Teppo Nieminen

Toiminnanohjausjärjestelmä verkkokaupan tueksi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

27.11.2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Teppo Nieminen Toiminnanohjausjärjestelmä verkkokaupan tueksi 43 sivua + 1 liitettä 27.11.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Yliopettaja Kari Aaltonen

Insinöörityössä toteutettiin yrityksen verkkokauppaan raportointijärjestelmä varaston ja myynnin seurantaan. Järjestelmä tuottaa myyntiraportteja ja varaston kehitykseen liittyviä tunnuslukuja, joita kuvataan visuaalisesti kaavioita ja taulukoita käyttäen. Työssä määritettiin prosessit Drupal-sisällönhallintajärjestelmää päälle pohjautuvan verkkokaupan raportoinnin kehittämiseen. Verkkokaupan alustana toimii Übercart, jonka alkuperäistä raportointia kehitetään eteenpäin omana moduulipakettina. Moduulipaketti sisältää toiminnanohjausjärjestelmistä tuttuja toimintoja mukautettuna verkkokauppaympäristöön.

Työn tavoite oli rakentaa verkkokauppaan aiempaa parempi raportointijärjestelmä käyttäen hyväksi erilaisia menetelmiä tiedon havainnollistamiseksi. Järjestelmän tuli soveltaa tuotteiden raportoinnissa ABC-analyysiä ja varastoseurantaa.

Toteutus tehtiin tiiviissä yhteistyössä yrityksen kanssa koko projektin ajan. Raportointijärjestelmä pohjautuu yrityksen tarpeisiin ja niihin pohjautuviin ratkaisuihin. Työssä tutkittiin Übercart-raportoinnin puutteita ja haettiin ratkaisuja puutteiden korvaamiseksi. Järjestelmän toteutuksessa käytettiin interaktiivisia kaavioita modernin HTML- ja CSS-määrittelyjen avulla.

Insinöörityön lopputuloksena syntyi järjestelmä, jonka avulla yrittäjä pystyy seuraamaan varastoa ja voi puuttua ongelmakohtiin välittömästi. Tehty moduulipaketti toimii myös alustana jatkokehitykselle, jossa on helposti mahdollista laajentaa toiminnollisuutta. Järjestelmän käyttöönoton vaikutusta myynnin kehitykseen ei pystytty työn puitteissa mittaamaan.

Järjestelmän toteutusta voi pitää onnistuneena ja tavoitteet saavutettiin. Työmäärällisesti räätälöidyn toiminnanohjausjärjestelmän rakentaminen saattaa olla valmiiden ratkaisujen integrointia huonompi, mutta insinöörityön puitteissa haluttiin selvittää, mikä olisi mahdollista tehdä käytössä olevaan verkkokauppaan.

Avainsanat	Drupal, Übercart, varastonhallinta, verkkokauppa

Author	Teppo Nieminen
Title	Enterprise resource planning as part of the e-commerce system
Number of Pages	43 pages + 1 appendices
Date	27 November 2015
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Kari Aaltonen, Principal Lecturer

The purpose of this bachelor's thesis is to develop a reporting system for tracking sales and stock development of the e-commerce system. The system includes sales reports and inventory management related key values used to measure performance of e-commerce and deliver viable information of e-commerce development. The reports include rich data visualization by interactive graphs and data tables. The system is based on Drupal content management system with Übercart e-commerce module. In this work Übercart reporting capabilities are extended by the set of the modules developed for their specific tasks (sales and stock).

The goal of this work is to help e-commerce seller to understand better sales development and what is happening in inventory. The created system needs to be able to deliver information of product and product group sales performances with numbers as well as a graphical representation. The seller needs to be able to identify badly performing products and be able to make justified decisions on stock purchases. The system makes use of the product key values such as ABC-analysis and inventory turnover ratio among other values.

The end result is fully functional reporting system where seller is able to comprehend all reports and make decisions based on reports and not just gut feeling. Created set of modules work as a foundation for future products to improve and automate e-commerce management even further. The impact of new reporting system to the normal business growth could not be measured reliably during this development time.

The completed project met the set requirements and overall project result is successful. The amount of time required to build enterprise resource planning system might be less beneficial than integrating ready system, but the point of this bachelor's thesis was to find out what is possible to do with the current e-commerce platform.

Keywords	Drupal, Übercart, inventory management, e-commerce

Sisällys

Lyhenteet

CMF	Content Management Framework. Ohjelmisto, jonka avulla voidaan rakentaa monipuolisia web-järjestelmiä tai sisällönhallintajärjestelmiä.
SQL	Structured Query Language. Tietokantakieli, jolla kommunikoidaan erilaisten SQL-pohjaisten tietokantapalvelimien kanssa.
CSV	Comma Separated Values. Tiedostomuoto, jolla tallennetaan taulukkotietoja tekstitiedostoon. Taulukon tietueet erotellaan erottimella, joka yleensä on pilkku, puolipiste tai sarkain.
FIFO	First-in, first-out. Menetelmä tai prosessi, jolla varastosta myydään tuotteen vanhin kappale ensin.

1 Johdanto

Insinöörityön tavoitteena on ratkaista erään verkossa toimivan pk-yrityksen kasvun mukanaan tuomat ongelmat resurssienhallinnassa. Yritys myy tuotteita kuluttaja-asiakkaille verkkokaupan avulla. Liiketoiminnan kasvun yhteydessä on yrityksen toiminnassa muodostunut ongelmaksi seurata myynnin ja varaston kehitystä kunnollisten seurantatyökalujen puutteen vuoksi.

Tavoitteena on suunnitella ja toteuttaa seurantatyökalut verkkokaupan toiminnan seuraamiseksi reaaliaikaisesti sekä kehittää verkkokauppaan myyntiä edistäviä toimintoja, kuten esimerkiksi automaattinen hinnoittelu. Tavoitteisiin ei kuulu täysiverisen toiminnanohjaus- tai asiakkuudenhallintajärjestelmän luominen, vaan suunnitella ja toteuttaa tarpeellisimmat toiminnot molemmista järjestelmistä helpottamaan verkkokaupan ylläpitoa ja siihen liittyviä tehtäviä.

Verkkokaupan arkkitehtuuri pohjautuu Drupal 7 -sisällönhallintajärjestelmään ja lisämoduuleihin, jotka lisäävät järjestelmän toiminnollisuutta. Työssä keskitytään suunnittelemaan ja kehittämään kolme moduulia, joiden avulla toteutetaan toiminnanohjausjärjestelmän perusta tiiviisti integroituna verkkokauppaan ja Drupal-järjestelmään käyttäen hyväksi tietokannan tauluja ja ohjelmointirajapintaa. Lisäksi työssä tutkitaan varastointiin liittyviä tunnuslukuja ja niiden käyttöä teoriassa ja käytännössä.

2 Raportointi kehittämään liiketoimintaa

2.1 Ongelmakohdat

Insinöörityön asiakasyrityksen verkkokaupassa on rajallisesti raportointityökaluja, eivätkä ne suoraan kerro liiketoiminnan kehityksestä tai suunnasta. Järjestelmässä on yksinkertainen myynti- ja varastoraportti, mutta ne ovat osoittautuneet riittämättömäksi ja vaativat lisää älykkäitä ratkaisuja. Yritys on erityisesti kiinnostunut seuraamaan varaston arvon kehitystä ja myyntiraportteja kolmella eri tasolla: tuotetasolla, tuoteryhmätasolla ja verkkokaupan kokonaiskuvassa. Lisäksi järjestelmän tulisi osata vertailla tuotteita ja tuoteryhmien suoriutumista annetuilla aikaväleillä.

Yrityksen ostohankinnat on tehty osaksi pohjautuen intuitioon ilman konkreettista tietoa perustelemaan hankinta. Saatetaan hankkia tuotteita, joiden nykyisessä varastonkierrossa on ongelma, tai pahimmillaan tuotetta tilataan lisää kahdesti, koska hankinnoista ei ole keskitetysti tietoa tallessa. Tuotteet saattavat olla varastossa pitkiäkin aikoja, koska tuotteella ei ole kysyntää. Lisäksi varastossa on tuotteita, joita ei ole myyty ainuttakaan kappaletta. Koska tärkeitä tietoja puuttuu, on yrityksen liiketoiminnan suunnan ja hankintojen ohjauksessa isoja ongelmia.

2.2 Varastonhallinta

Järjestelmässä varastoa seurataan tarkkaillen tuotteiden ja tuoteryhmien menekkiä käyttäen apuna erilaisia tunnuslukuja. Tällaisia tunnuslukuja ovat esimerkiksi myyntimäärä, katetuotto, kateprosentti, ABCD-analyysin luokitus ja varaston kiertonopeus. Tunnuslukujen avulla pystyy kartoittamaan helposti tuotteiden tilanteen verrattuna vastaaviin tuotteisiin.

Katetuotto lasketaan yksinkertaisella kaavalla 1.

$$Katetuotto = myynti - ostot \quad (1)$$

Myynti on tuotteen veroton myyntihinta.
Ostot ovat tuotteen veroton ostohinta.

Katetuottoprosentti lasketaan kaavalla 2.

$$Katetuottoprosentti = \frac{katetuotto}{myynti} \times 100 \quad (2)$$

ABCD-analyysi

Yksi tärkeä tunnusluku tuotteiden ja tuoteryhmien suorituskykymittauksessa on ABCD-analyysi tai tunnetummin ABC-analyysi. Analyysillä erotellaan varaston tuotteet luokkiin arvon ja menekin mukaan. Analyysi perustuu Pareton periaatteeseen, jossa 20 % tuotteista vastaa 80 %:a myynnistä tai varastoarvosta. Periaate tunnetaan myös nimellä ”80–20-sääntö”. Periaate sai alkunsa, kun Joseph Juran löysi Vilfredo Pareton tutkimuksen, jossa ilmeni, että 20 % väestöstä omistaa noin 80 % maan varoista [1, s. 282]. Juran huomasi samankaltaisen säännön ilmenevän useissa asiayhteyksissä ja nimesi ilmiön Pareton mukaan Pareton periaatteeksi. [2, s. 62; 3.]

ABC-analyysissä ei ole ehdotonta sääntöä tai luokkien määrää, vaan sääntöjä sovelletaan käyttökohteiden ja tarpeiden mukaisesti. Luokittelun tulos on suuntaa antava, eikä tieto itsessään tehosta liiketoimintaa, mutta tuloksen pohjalta voi lähteä tekemään tarpeellisiksi näkemiään toimenpiteitä. Analyysi ei myöskään kerro yksittäisen tuotteen tärkeyttä tuotekokonaisuudessa, koska B- tai C-luokan tuotteella voi olla merkittävä rooli A-luokkaan kuuluvien tuotteiden myynnissä. Luokituksella ei siis saa selville tuotteiden vaikutuksia toisiin tuotteisiin. [2, s. 63–64.]

Järjestelmässä käytetyn ABCD-luokittelun säännöt ovat seuraavat:

- Tuotteet järjestetään katetuoton mukaan laskevaan järjestykseen.
- A-luokkaan sisältyvät 75 % kokonaiskatetuotosta tehneet tuotteet.
- B-luokkaan sisältyvät seuraavat 20 % kokonaiskatetuotosta tehneet tuotteet.
- C-luokkaan sisältyvät jäljelle jäävät tuotteet.

- D-luokkaan sisältyvät tuotteet, joiden katetuotto on ollut mitatulla aikavälillä nolla tai negatiivinen.

D-luokassa olevat myymättömät ja tappiolla myydyt tuotteet olisi vielä mahdollista luokitella kahteen osioon, jolloin saisi eroteltua tarkemmin ne tuotteet, joita on myyty tappiolla.

Analyysiä on mahdollista soveltaa tuotteisiin, jolloin saa helposti selville, miten tuotteet suoriutuvat keskenään, ja saadaan selville tuotteet, joiden varastomääriä tulisi valvoa tarkemmin. Luokittelulla saadaan myös selville tuotteet, joiden varastonkierrossa on ongelma tai joiden menekki on muuttunut. Tulokset ovat riippuvaisia aina mitatusta aikavälistä, ja esimerkiksi C-luokan tuote saattaa olla toisella aikavälillä A-luokassa. Tuotteen saatavuus vaikuttaa myös luokitukseen, koska katetuottoa ei synny tuotteista, joita ei voi myydä kysynnästä huolimatta, koska varastoa ei ole. [5, s. 21; 1, s. 283.]

Tuotteiden myyntinopeus vaihtelee. Edulliset tuotteet liikkuvat nopeammin ja kalliimmat hitaammin. Edullisissa tuotteissa on pienempi katetuotto ja vastaavasti kalliimmissa on isompi. Katetuotolla tehty ABCD-analyysi tarjoaa tasavertaisemman tuloksen, kuin esimerkiksi myyntimääriin perustuva analyysi.

Varaston kiertonopeus

Varaston kiertonopeudella selvitetään, kuinka varaston tuotteet kiertävät varastossa. Kiertonopeudella voidaan mitata, kuinka monta kertaa tuote myydään ulos varastosta annetulla aikavälillä tai montako päivää kuluu keskimäärin myydä tuote ulos varastosta. Varaston kiertonopeudella tavoitellaan mahdollisimman suurta lukua, mikä tarkoittaa varastoon sidotun pääoman vapautumista nopeasti. Kiertoaikaa voidaan mitata erilaisilla aikaväleillä, mutta vertailun helpottamiseksi verrattavien aikavälien tulisi olla pituudeltaan vastaavanlaiset. Varastonkiertoa mitataan kaavan 3 mukaisesti. [5, s. 18; 6; 7.]

$$Varaston\ kiertonopeus = \frac{myytyjen\ tuotteiden\ kustannus}{varaston\ keskiarvo} \qquad (3)$$

Myytyjen tuotteiden kustannukset sisältävät tuotteen hankintamenot. Varaston keskiarvo on laskettu keskiarvo tarkasteltavalla aikavälillä.

Kaavasta saadulla varaston kiertonopeuden arvolla voi selvittää kiertonopeus päivinä käyttämällä kaavaa 4.

$$Varaston\ kiertonopeus\ p\ddot{a}ivin\ddot{a} = \frac{p\ddot{a}iv\ddot{a}t}{varaston\ kiertonopeus} \qquad (4)$$

Päivät ovat tarkasteltavan aikavälin päivien lukumäärä.

Varaston kiertonopeudella ja ABCD-analyysillä saadaan selville tuotteen menekki, ja tietojen pohjalta voidaan tehdä konkreettisempia päätöksiä varastohankintoihin. Tunnusluvut eivät ennusta tulevaisuutta, mutta muutoshistoriasta saa käsityksen tuotteen käyttäytymisestä ja mahdollisista trendeistä erilaisilla ajanjaksoilla.

2.3 Varastonhallinta osana toiminnanohjausjärjestelmää

Toiminnanohjausjärjestelmä on kokonaisuus tai keskusjärjestelmä yrityksen liiketoimintaan liittyvien toimielimien integrointiin. Järjestelmässä kaikkien yrityksen toimielimien tieto säilytetään keskitetysti, ja se on sitä kautta kaikkien sitä tarvitsevien ulottuvilla. Esimerkiksi järjestelmästä löytyvät asiakkaan tekemät tilaukset ja niihin liittyen tieto toimitetuista tuotteista ja tieto laskutuksesta. Tällaiset tiedot eivät välttämättä aina löydy samasta järjestelmästä. Tietojen etsiminen useista järjestelmistä lisää selvitystyötä ja kasvattaa virheiden mahdollisuutta. [8.]

Toiminnanohjausjärjestelmässä varastonhallinta tarkoittaa varastonohjausta. Yhtenä komponenttina voi olla esimerkiksi varastokirjanpito, joka on kehityksen ja reaaliaikaisen tilanteen seurantaa. Järjestelmästä ilmenevät artikkelien hankintamenot, myyntimäärät ja varastomäärät takautuvasti. Toiminnanohjauksessa varastonhallinta toimittaa varastokirjanpitoa, jonka tietoja muut toiminnanohjausjärjestelmän osat hyödyntävät raporteissa ja analyyseissä. [9, s. 71–73.]

3 Verkkokauppajärjestelmän arkkitehtuuri ja suunnittelu

Raportointijärjestelmän suunnittelu alkaa yrityksen ongelmakohtien ja tarpeiden määrittelyllä. Selkeimmät teemat ovat tiedon visualisointi, tunnuslukujen käyttö ja helppokäyttöisyys. Vaatimuksien perusteella järkevintä on lähteä toteuttamaan moduulikokonaisuutta, jossa päämoduuli sitoo alamoduulien toiminnot yhteen selkeästi osioiduksi kokonaisuudeksi. Kaikissa moduuleissa yksi edellytys on saada katsottua tietoja takautuvasti asetetulla aikavälillä, joka voi olla päivästä useisiin vuosiin.

3.1 Drupal lyhyesti

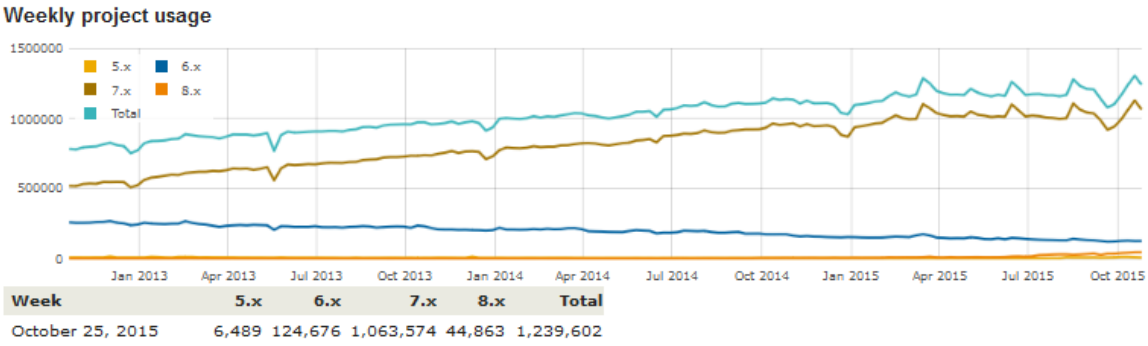
Drupal on PHP-kielellä kirjoitettu avoimen lähdekoodin sisällönhallintajärjestelmä tai CMF (Content management framework). Drupal mahdollistaa suurien ja monimutkaisten verkkosivujen toteutuksen ja tarjoaa erittäin kattavan ohjelmointirajapinnan. Järjestelmän arkkitehtuuri on modulaarinen, joten toimintoja pystyy vaivattomasti lisäämään asentamalla haluttuja toimintoja tarjoavan moduulin tai tarvittaessa luomalla moduulin, joka toimii itsenäisesti tai laajentaa jonkin muun moduulin toimintaa.

Järjestelmävaatimuksiltaan Drupal 7 vaatii toimiakseen seuraavanlaisen ympäristön:

- Apache-, Nginx- tai Microsoft IIS -web-palvelin
- Tuetut tietokantapalvelimet
    - MySQL:n versio 5.0.15 ja uudemmat, edellyttää PDO-ajurin
    - PostqreSQL:n versio 8.3 ja uudemmat, edellyttää PDO-ajurin
    - SQLiten versio 3.3.7 ja uudemmat
    - Microsoft SQL ja Oracle tuettuina lisämoduulin avulla
- PHP:n versio 5.2.5 tai uudemman asennus web-palvelimeen [10].

Muistivaatimus Drupal 7:n ydinasennuksella on 32 Mt, mutta sivustokohtaisesti ja asennetun moduulimäärän mukaan muistitarve kasvaa. Monimutkaisissa ja suurissa sivustoissa muistia ja laskentatehoa vaaditaan palvelimelta, jotta sivuston toimii sulavasti. [10.]

Maailmanlaajuisesti Drupal-pohjaisia verkkosivustoja on noin 1,24 miljoonaa. Luku on vain suuntaa antava, koska tulos perustuu sivustoihin, joissa on käytössä vapaaehtoinen päivitysmoduuli. Määrä on kuitenkin nouseva uusimman vakaan ytimen kohdalla, kuten kuvasta 1 voidaan nähdä version 7.x jakelun kehityksessä. [11.]



Kuva 1. Drupal-järjestelmän asennusmäärä pääversion ja kokonaismäärän mukaan [11].

Järjestelmän sisältötyypeillä luotua sisältöä kutsutaan nimellä Node (Solmu). Termeillä ei ole vakiintunutta suomennosta, joka avaisi termin tarkoitusperää. Node-sisältöä ovat esimerkiksi sisältösivut tai verkkokaupan tuotteet. Tällaisilla sisällöillä on uniikki node ID -tietue, jolla yksilöidään sisällöt ja järjestelmä pystyy hakemaan sisältöä. [12; 13.]

Toinen tärkeä termi on Entity (Entiteetti). Entiteetillä kuvataan tietomallia. Entiteetin rakenne voi vaihdella sen tyypin mukaisesti. Tyypillisin entiteettityyppi on *node*. Yksi entiteetti voi sisältää useita omia entiteettejä. [14.]

Keskeistä on myös tietää, miten hook-järjestelmä (Koukku) toimii. Koukut ovat pääasiallinen tapa vuorovaikuttaa Drupalin ja moduulin välillä. Koukkuja on paljon erilaisia monenlaisiin tilanteisiin. Esimerkiksi *hook_menu*-koukulla voi moduuli esittää omia URL-polkujaan ja niihin liittyviä suoritettavia funktioita. Moduulit voivat kytkeytyä koukkuihin käyttämällä nimeämiskäytäntöä *omaModuuli_koukunNimi*. Esimerkiksi *hook_menu*-koukun käyttö olisi muotoa *omaModuuli_menu*. Drupal kutsuu näitä koukkuja automaattisesti suorituksen aikana eri vaiheissa. [15; 16.]

3.2 Järjestelmän arkkitehtuuri

Verkkokaupan toiminnollisuus on tuotu Drupal-alustaan aiemmin käyttäen Übercart-nimistä moduulia. Se sisältää kaikki verkkokaupan peruselementit, kuten ostoskorin, kassan, tilaushallinnan ja tuotepohjan. Lisäksi käytettävissä on erilaisia lisämoduuleja, joilla verkkokauppaa saa räätälöityä tarpeiden mukaisesti.

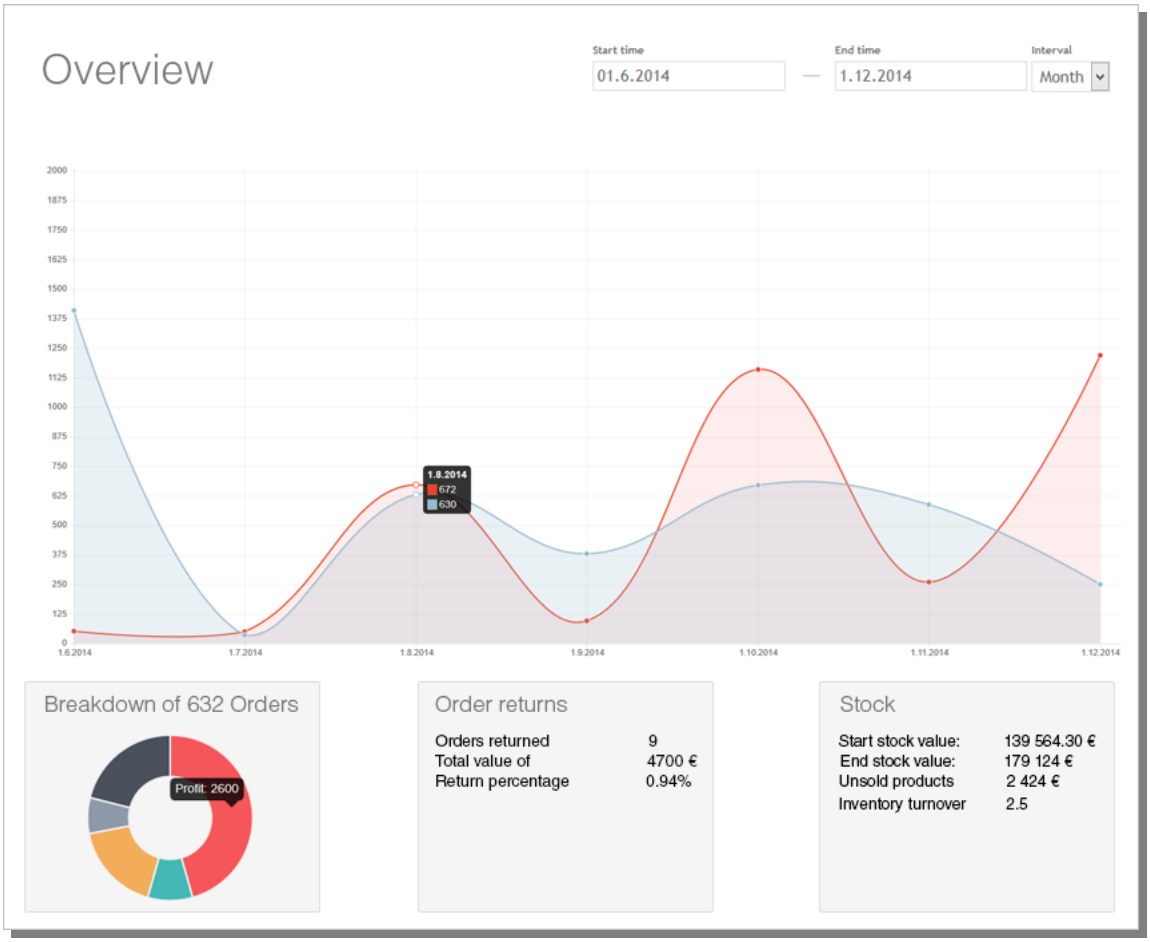
Työn keskeisimpinä osina ovat Übercart-moduulin tietokannan taulut, joista noudetaan tietoa raporttien ja toimintojen rakentamiseksi. Taulukossa 1 on listattuna Übercart-moduulin keskeisimmät tietokannat ja niiden tarkoitus.

Taulukko 1. Übercart-moduulin keskeiset tietokantataulut.

Taulun nimi	**Tarkoitus**
uc_products	Kaupan tuotteet ja niiden tuotekohtaiset tiedot, kuten ostohinta, myyntihinta, tuotekoodi ja lisäksi tuotekohtaiset mitat ja paino. Tuotteet ovat nodeja, joten taulusta löytyy myös node ID -tietue, jota voidaan käyttää tuotteen yhdistämiseksi tuotesisältöön.
uc_orders	Kaikki tilaukset ja tilaukseen liittyvät tiedot. Tietoihin lukeutuu tilauksen kokonaishinta, toimitustiedot, tuotemäärä, maksutapa, tilauksen tila, luonti ja päivitysajankohta.
uc_order_products	Tiedot tilauksen tuotteista. Tilaushetken sisäänostohinta, myyntihinta, tuotemäärä ja tuotteen nimi tallentuvat tilauskohtaisesti tilausnumeron mukana tauluun.
uc_order_line_items	Tilausta koskevat postitus- ja verotustiedot. Taulussa on tilauskohtaisesti tilauksesta aiheutuvat verokustannukset senhetkisen ALV-määrän mukaisesti sekä postimaksun määrä. Myös tilauksessa käytetyt kupongit tallentuvat tauluun.
uc_product_stock	Tuotteiden nykyhetkinen varastotilanne jokaiselle tuotteelle tuotenumeron ja node ID:n kanssa.

3.3 Raportointityökalujen suunnitelma

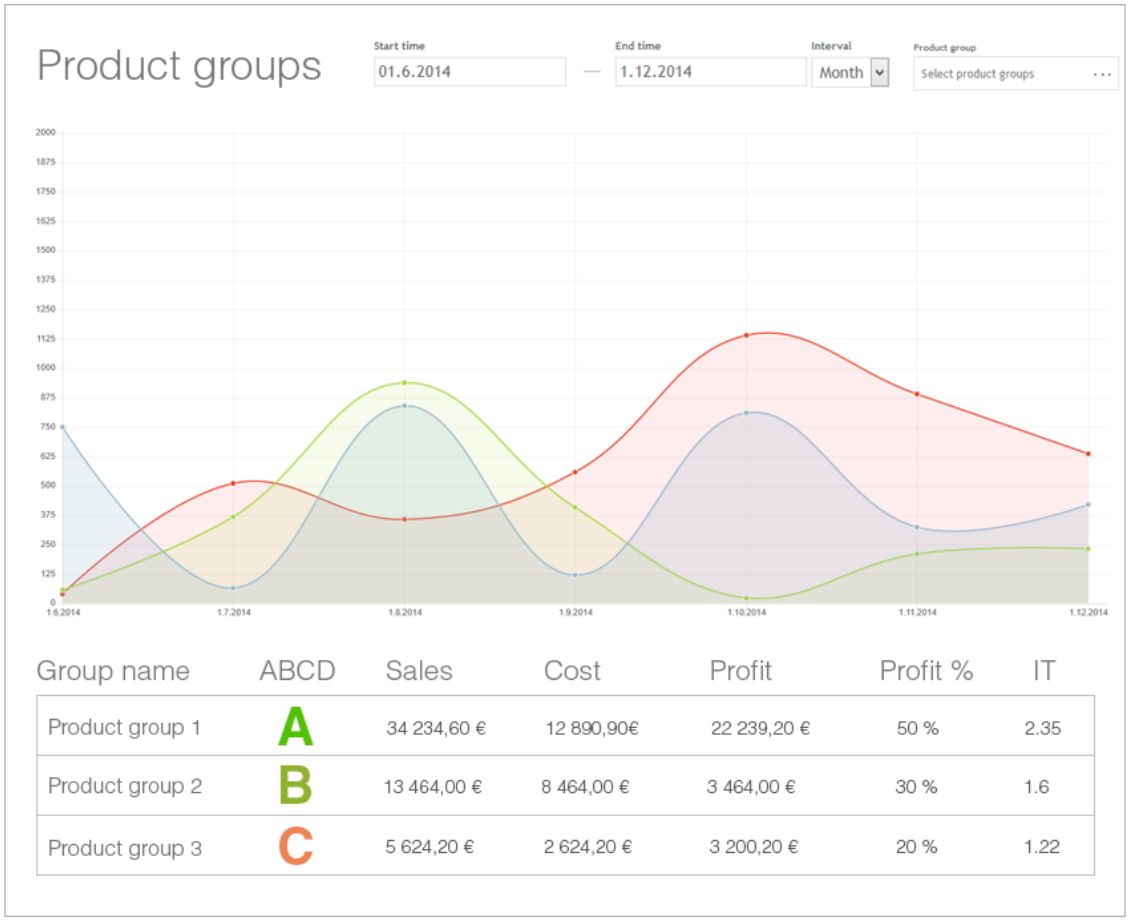
Raportointijärjestelmän päämoduulin on tarkoitus tarjota kokonaisuudelle pohja, jonka päälle rakentuvat raportointityökalut. Moduulin tulisi näyttää lukijalle tärkeimmät myyntiraportit jaoteltuna kolmelle sivulle. Tarkoituksena on näyttää verkkokaupan ylläpitäjälle ensimmäisellä sivulla kaupan kehitys ja yleiskatsaus annetulla aikavälillä. Kaupan kehitystä visualisoidaan näyttämällä myynnistä grafiikka asetetulle aikavälille sekä tehtyjen tilausten tuotot, menot ja mahdollisten palautettujen tilauksien määrä. Kuvassa 2 on havainnollistettu pääsivun rakennetta. Sivulla tulee siis olla kaikki olennainen tieto yleistasolla, josta on mahdollista selvittää liiketoiminnan kehitys nopeasti.



Kuva 2. Hahmotelma raporttijärjestelmän pääsivun raportista, jossa näytetään tiedot valitulta aikajaksolta.

Toisen sivun tarkoitus on näyttää visuaalisesti ja numeroin valittujen tuoteryhmien suoriutuminen keskenään asetetulla aikavälillä. Tuoteryhmille lasketaan tunnusluvut ja tehdään ABCD-analyysi, jolloin tuoteryhmien erot havaitaan helposti. Tunnuslukujen tueksi näytettävä grafiikka auttaa lukijaa ymmärtämään, miten tunnusluvut ovat rakentuneet, ja

ennen kaikkea havainnollistamaan trendejä, joita ei välttämättä suoraan numeroista löytäisi helposti. Kuvassa 3 on hahmotelma tuoteryhmäraportista ja raporttiin liittyvistä tiedoista, kun ylläpitäjä on valinnut vertailtavat tuoteryhmät.



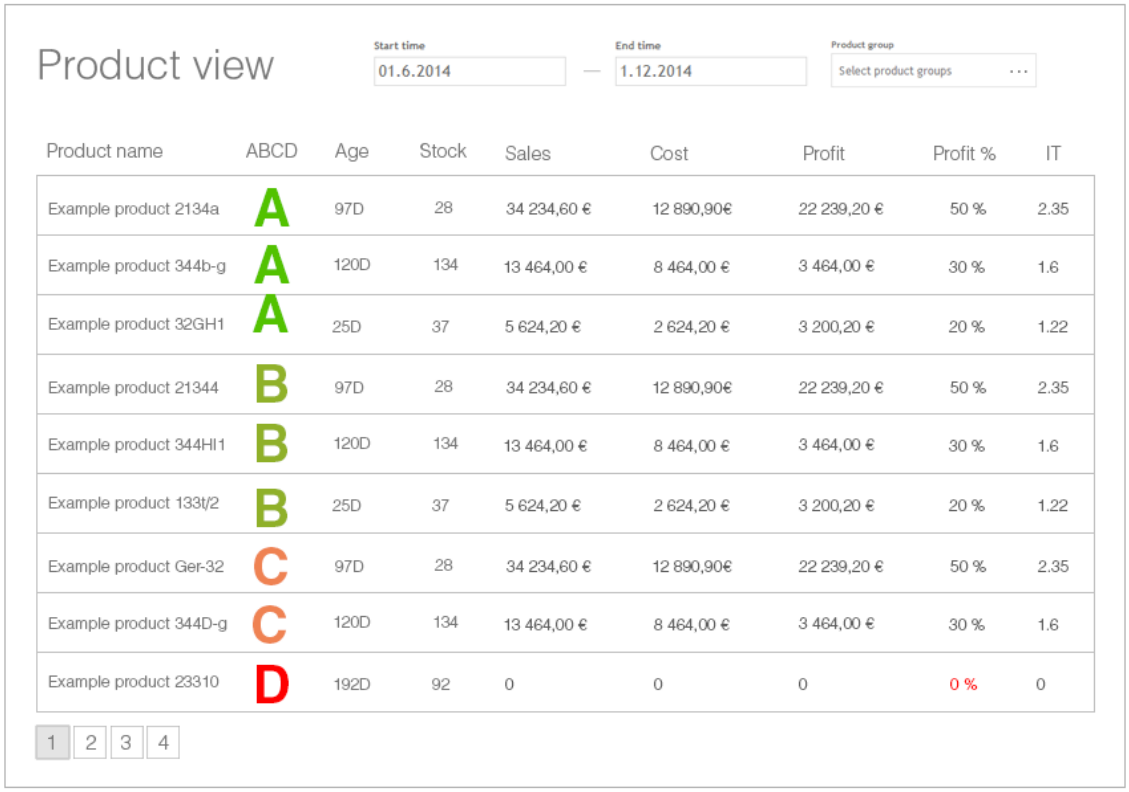
Group name	ABCD	Sales	Cost	Profit	Profit %	IT
Product group 1	A	34 234,60 €	12 890,90€	22 239,20 €	50 %	2.35
Product group 2	B	13 464,00 €	8 464,00 €	3 464,00 €	30 %	1.6
Product group 3	C	5 624,20 €	2 624,20 €	3 200,20 €	20 %	1.22

Kuva 3. Tuoteryhmien hahmotelma, jossa valitut ryhmät on kuvattu grafiikan ja taulukoiden avulla.

Kolmas sivu sisältää valittuihin tuoteryhmiin kuuluvien tuotteiden vertailun. Tuotetasolla näytetään listaus kaikista tuotteista, jotka kuuluvat ryhmään, ja niille lasketut tunnusluvut sekä ABCD-luokka. Tuotetasolla lukijan tulisi nähdä selkeästi, miten tuotteet menestyvät verrattuna toisiinsa, ja pystyä seuraamaan reaaliaikaisesti tuotteiden varastotilannetta. Sivun tarkoituksena olisi antaa selkeä kuva varastotarpeesta. Ongelmalliset tuotteet on mahdollista löytää asettamalla pitkä aikaväli ja tarkastelemalla D-luokan tuotteita, jotka sitovat yrityksen pääomaa varastoon pitkiksi ajoiksi.

On mainittu, että ABC-analyysiä ei tulisi käyttää tuoteryhmiin, mutta tehtävässä järjestelmässä kokeillaan, kuinka analyysi toimii tuoteryhmien vertailussa [5, s. 21]. Lähtökohtaisesti kaikki järjestelmän tuoteryhmien tuotteet ovat hyvin homogeenisiä, joten vertailun tuoteryhmätasolla tulisi onnistua. Analyysin tekeminen vaatii kuitenkin, että otanta on tarpeeksi suuri.

Tuotelistauksessa on paljon tärkeää tietoa, ja sen graafinen esittäminen ei ole tuotemäärien puolesta mielekästä. Kuvassa 4 on tuotelistauksen hahmotelmaa ja sen sisältämiä tuotteisiin liittyviä tietosarakkeita. Tuotelistauksen pituus voi ylläpitäjän valitsemien tuoteryhmien mukana kasvaa useisiin sivuihin, joten taulukon tulisi käyttää sivutusta ja näyttää yhdellä sivulla sama määrä tuotteita. Taulukko on sisältömäärälle luonnollinen esitystapa, koska taulukkoa voi järjestää osioiden mukaan nousevaan tai laskevaan järjestykseen.



Kuva 4. Tuotetason hahmotelma, jossa valittujen ryhmien tuotteet ovat vertailussa.

Tuotteen ikä (montako päivää tuote on ollut järjestelmässä) on syytä listata tuotteita verrattaessa, koska uusi tuote ei välttämättä saa hyvää ABCD-analyysin luokitusta, koska myyntiaika on lyhyt ja se heijastuu katetuottoon. Uusien tuotteiden luokituksen pystyisi

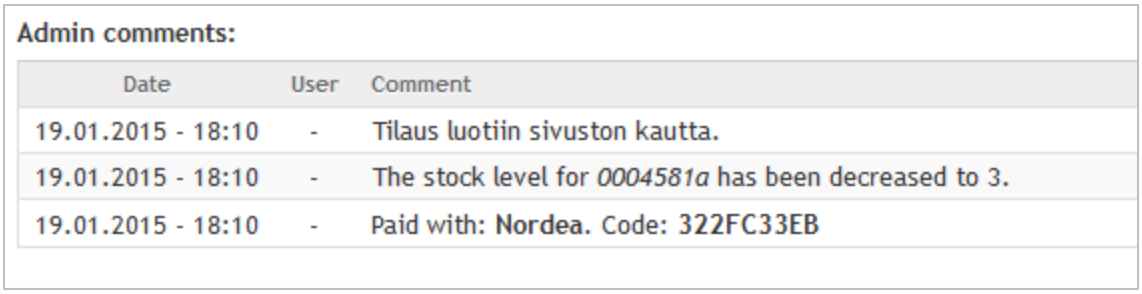
esimerkiksi tarkistamaan lyhentämällä aikaväliä, jolloin uusi tuote voidaan mitata koko tarkasteluaikavälillä ja saada luotettavampi luokitus.

3.4 Varastohallinnan suunnitelma

Insinöörityössä tehtävällä varastomoduulilla on tarkoitus paikata järjestelmän puuttuva varastohistoria ja kerätä varastohankinnat talteen keskitetysti. Lisäksi jatkossa tapahtuvat varastomuutokset tulisi saada tallennettua tietokantaan aikajärjestyksessä. Nämä toiminnot on toteutettava omana moduulinaan, koska Drupal-sivustolla ei tämänkaltaisia toimintoja tarjoavia lisämoduuleja ole.

Varastohankintojen kerääminen onnistuu luomalla Drupal-sisältötyyppi, joka sisältää tiedot tulevista tilauksista. Varastohankinnoista tulee kerätä talteen tieto hankintamääristä, hankintahinnasta, hankintapäivämäärä ja tuotteen saapuneeksi kuittaava valintakenttä. Tuotetäydennyksen saapuessa voisi sisältöä verrata aiemmin tehtyyn tilaukseen ja merkitä tuote saapuneeksi, jolloin järjestelmä voi automaattisesti kasvattaa tuotteen varastomäärää tilaukseen merkityllä määrällä. Tilauslomakkeen helppokäyttöisyyden kannalta on järkevää lisätä mahdollisuus täydentää tilaustiedot käyttäen esimerkiksi CSV-tiedostoa, jolloin tilattavien tuotteiden koostaminen onnistuu esimerkiksi Microsoft Excel -ohjelmalla. Sisältötyyppi on luonnollinen valinta tarpeen mukaisen lomakkeen ja tiedon tallentamiseksi, koska kaikki luodut varastotilaukset ovat silloin *node*-entiteettejä, jolloin Drupal hoitaa automaattisesti tietojen tallentamisen ja muokkaamisen.

Varastohistorian luominen on selkeästi vaikeampi toimenpide. Vaikka tilauksista saisi käänteisesti luettua varastomenekin, se ei kuitenkaan kerro varastotäyttöjen ajankohtaa tai tuotteiden varastokehitystä niiden tilaamattomien tuotteiden kohdalla. Tilauksista kuitenkin löytyy kuvan 2 (s. 9) mukaiset automaattisesti luodut ylläpitäjien kommentit, joissa järjestelmä kertoo tilauksen yhteydessä tehdyn varastomuutoksen tilatuille tuotteille. Kommenttikentän tiedoista pystyy rakentamaan historian hyvin lähelle totuutta. Varastotäydennyksien ajankohdat pystyy arvioimaan tilauksien välissä tapahtuneella tuotteen varastosaldon kasvulla.



Admin comments:

Date	User	Comment
19.01.2015 - 18:10	-	Tilaus luotiin sivuston kautta.
19.01.2015 - 18:10	-	The stock level for *0004581a* has been decreased to 3.
19.01.2015 - 18:10	-	Paid with: **Nordea**. Code: 322FC33EB

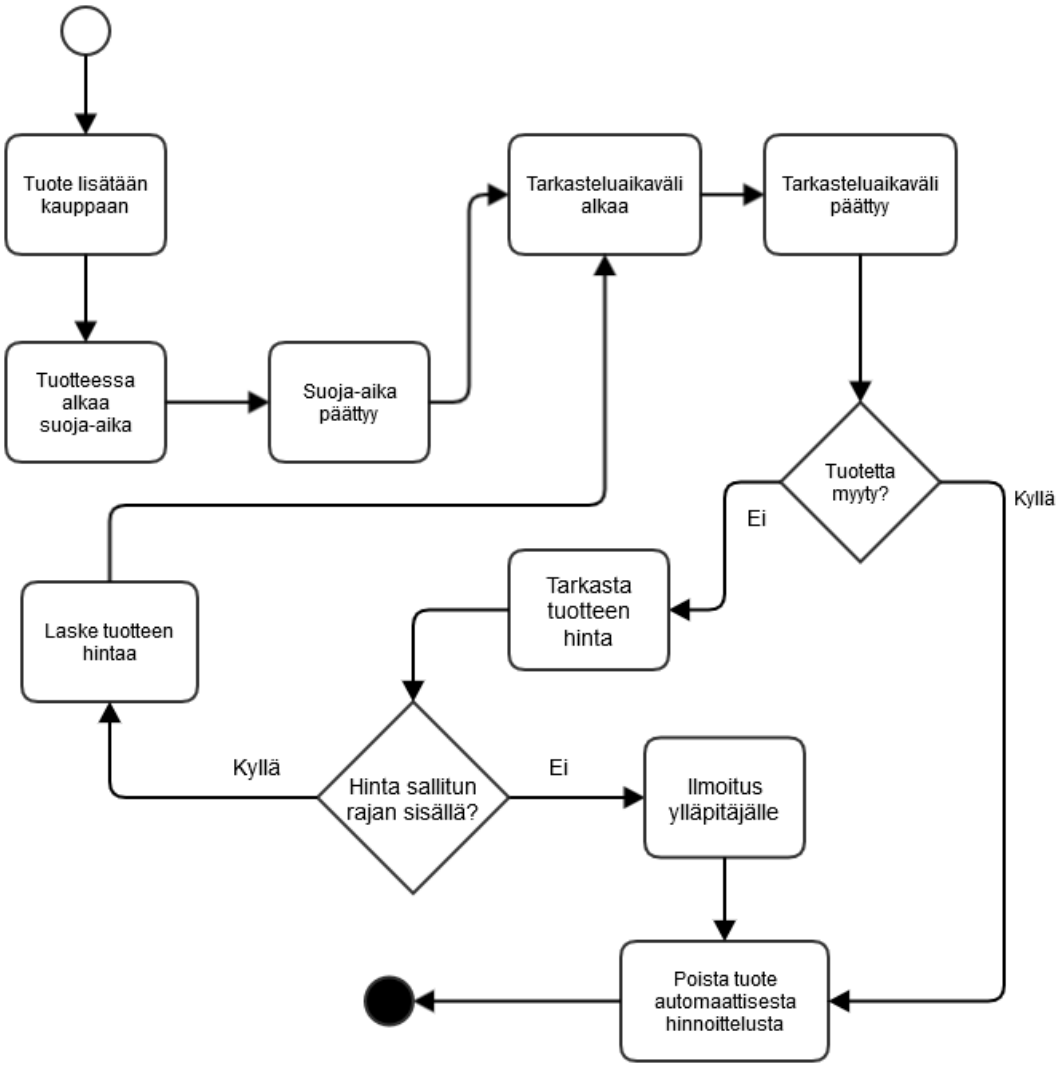
Kuva 5. Tilauksen ylläpitäjän kommentit, joissa varastotilanteen muutos on ilmoitettu.

Varastohistorian luonti onnistuu käymällä läpi kaikki tilauskommentit aikajärjestyksessä ja luomalla tilaushetken varastotilanteesta merkintä. Varastossa löytyvien myymättömien tuotteiden varastotilannetta ei tilauskommenteista löydy, mutta niissä voi olettaa tuotteen luontihetkellä olevan varaston olevan sama kuin nykytilanteessa on.

Varastotilanteen muutoksien seurantaan tulisi kehittää työkalu, joka integroituu Übercart-varastomoduulin toimintalogiikkaan tarkkailemaan muutoksia. Drupal mahdollistaa koukkujen tekemisen järjestelmään, ja niitä kutsutaan tietynlaisten toimenpiteiden aikana. Esimerkiksi Übercart voi kutsua kaikkia tuntemiaan koukkuja varastopäivityksen yhteydessä, ja ne voivat vuorostaan tehdä omia muutoksiaan tai yliajaa varastonpäivityskomennon. Koukun kautta olisi mahdollista tehdä varastomuutoksiin kuuntelija, joka sitten tekisi muutoksesta omaan tietokantatauluun historiamerkinnän.

3.5 Automaattinen hinnoittelu

Automaattisella hinnoittelulla voidaan edesauttaa ongelmallisten tuotteiden myyntiä antamalla järjestelmän kontrolloida hintoja. Hinnoittelun tulisi perustua sääntöihin ja ehtoihin, joiden pohjalta hinnoittelualgoritmi toimii. Ensisijaisesti automatiikan tulisi ohjata myyntitapahtumattomia tuotteita. Tällaisien tuotteiden kohdalla on vaikea lähteä rakentamaan hienojakoisempaa hinnanohjausta, joka perustuisi tuotteen myyntihistoriaan. Myymättömien tuotteiden hinnoittelualgoritmin tulisi noudattaa kuvan 6 kaavaa.



Kuva 6. Myymättömien tuotteiden käsittely luotavassa järjestelmässä.

Oletuksena kaikki kaupassa olevat tuotteet kuuluvat automaattisen hinnoittelun piiriin, mutta mahdollisuutena tulisi olla suojata tuote tai tuoteryhmä eli lisätä tuote listaan, joka on rajattu automatiikan ulkopuolelle. Kuvassa 6 mainitulla suoja-ajalla vaikutetaan tuotteen alkuhetkellä olevaan huonoon myyntiin, koska tuotteella menee hetki listautua hakukoneiden hakutuloksissa.

Hinnoittelussa tarkasteluaikavälin tulee olla tuotteen hinnasta riippuvainen, koska edullisempia tuotteita myydään nopeammin ja kalliimpia hitaammin. Hinnoittelussa tulee myös huomioida asetetut rajat, joiden ulkopuolelle automatiikka ei aseta hintaa. Tuotteen saavuttaessa hinnoittelun äärirajan hintaa ei enää muuteta. Äärirajan ylityksen jälkeen yllä-

pitäjälle lähetetään ilmoitus, jolloin vain manuaalisilla toimenpiteillä voi vaikuttaa tuotteeseen. Myymättömien tuotteiden automaattisen hinnoittelun prosessi päättyy, kun tuotteella on myyntitapahtuma tai hinnan ääriraja on saavutettu.

Alennettu hinta viestitetään sivuston käyttäjille tuotelistauksessa ja tuotesivulla näyttämällä myyntihinnan vieressä vanha hinta ja alennuksen suuruus prosentteina. Kuvassa 7 on esimerkki, kuinka alennetut tuotteet korostuvat tuotelistauksen tuotteista alennusprosentin ja vanhan hinnan avulla.

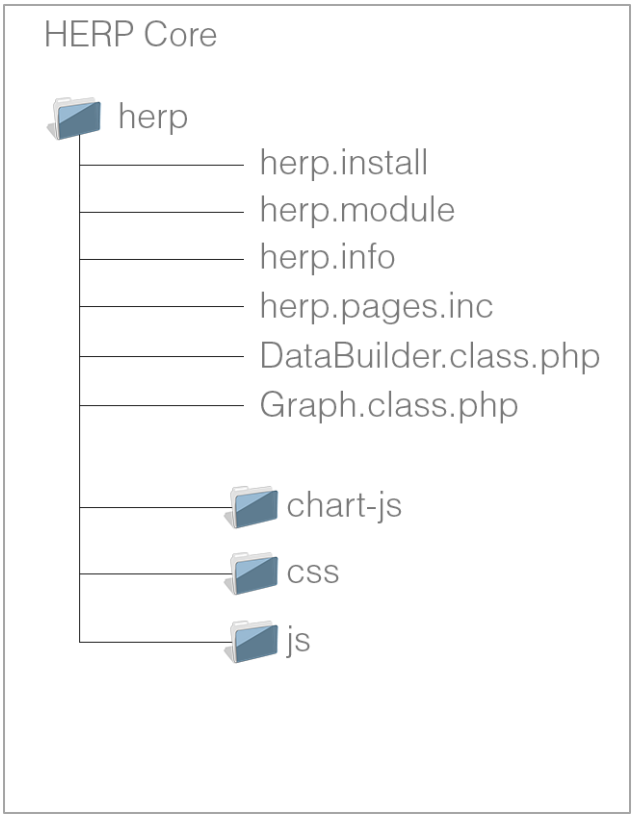
Kuva 7. Alennettujen tuotteiden näyttäminen tuotelistauksessa.

4 Raportointijärjestelmän toteutus

Luvussa 3 esitettyjen suunnitelmien pohjalta järjestelmän toteutus oli selkeää lähteä aloittamaan. Ensimmäinen tehtävä oli luoda moduulien pohjat Drupal-ympäristöön ja kehittää paketille yksilöllinen nimi. Lopulta kokonaisuuden nimeksi valikoitui *Horizon Enterprise Resource Planning*, lyhyesti HERP. HERP sisältää kolme moduulia: Core, Stock ja Price, jotka hoitavat oman toimialueensa tehtäviä. Nimet ovat tärkeät, koska niillä määritellään moduulien funktioiden nimeämiskäytännöt ja ne tekevät Drupal-järjestelmälle mahdolliseksi tunnistaa moduulin esittelemät koukut, joita käytetään osana moduulia.

4.1 Järjestelmän perusta

Raportoinnin päämoduulissa eli raportointityökalujen perustassa ovat kaikki Drupal-moduulin luomiseen tarvittavat tiedostot. Kuvassa 8 on puurakenteena kuvattu tärkeimmät moduulin käyttämät tiedostot. Drupal edellyttää *moduulin_nimi.info*-tiedostoa, jossa moduuli esittelee itsensä ja kertoo mahdollisista moduuliriippuvuuksista. Moduuliriippuvuuksilla varmistetaan, että moduulia ei voi ottaa käyttöön, ennen kuin vaaditut lisämoduulit on asennettuna ja otettuna käyttöön. Tällaisia päämoduulissa määritettyjä riippuvuuksia ovat Übercart, Date API ja Date picker. Date API tarjoaa työkalut päivämäärien käsittelyyn ja mahdollistaa Date picker -moduulin käytön, joka puolestaan tarjoaa käyttöön selaimissa JavaScript-skriptauskielellä toteutetun päivämäärävalitsimen kalenterinäkymässä.



Kuva 8. Raportoinnin Core-moduulin kansiorakenne.

herp.module on moduulin päätiedosto, jonka Drupal suorittaa toimintojen aikana. Tiedostossa tulisi olla sisällä kaikki moduulin käyttämät koukut ja funktiot, ellei tiedoston sisällä ole erikseen käytetty tapoja toisen tiedoston sisällyttämiseen, kuten PHP:n require tai include -funktioilla.

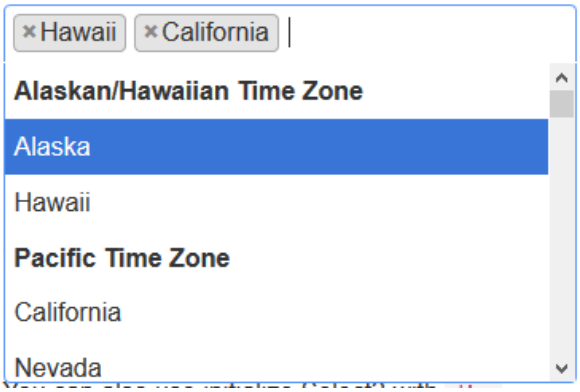
herp.install sisältää moduulin asennus- ja päivitysvaiheessa suoritettavia koukkuja, kuten tietokannan uusien taulujen luonnin tai uusien sisältötyyppien luonnin.

herp.pages.inc sisältää päämoduulin URL-polkujen kutsufunktioiden koodit, jotka on siirretty *herp.module*-tiedostosta. Moduulia on järkevää pilkkoa pienempiin ja kuvaaviin tiedostoihin, jolloin moduulin rakenne on helpompi ylläpitää.

DataBuilder.class.php on DataBuilder-luokan tiedosto ja varsinainen moduulin ydin, joka hakee ja tulostaa raportointisivut annettujen parametrien mukaisesti.

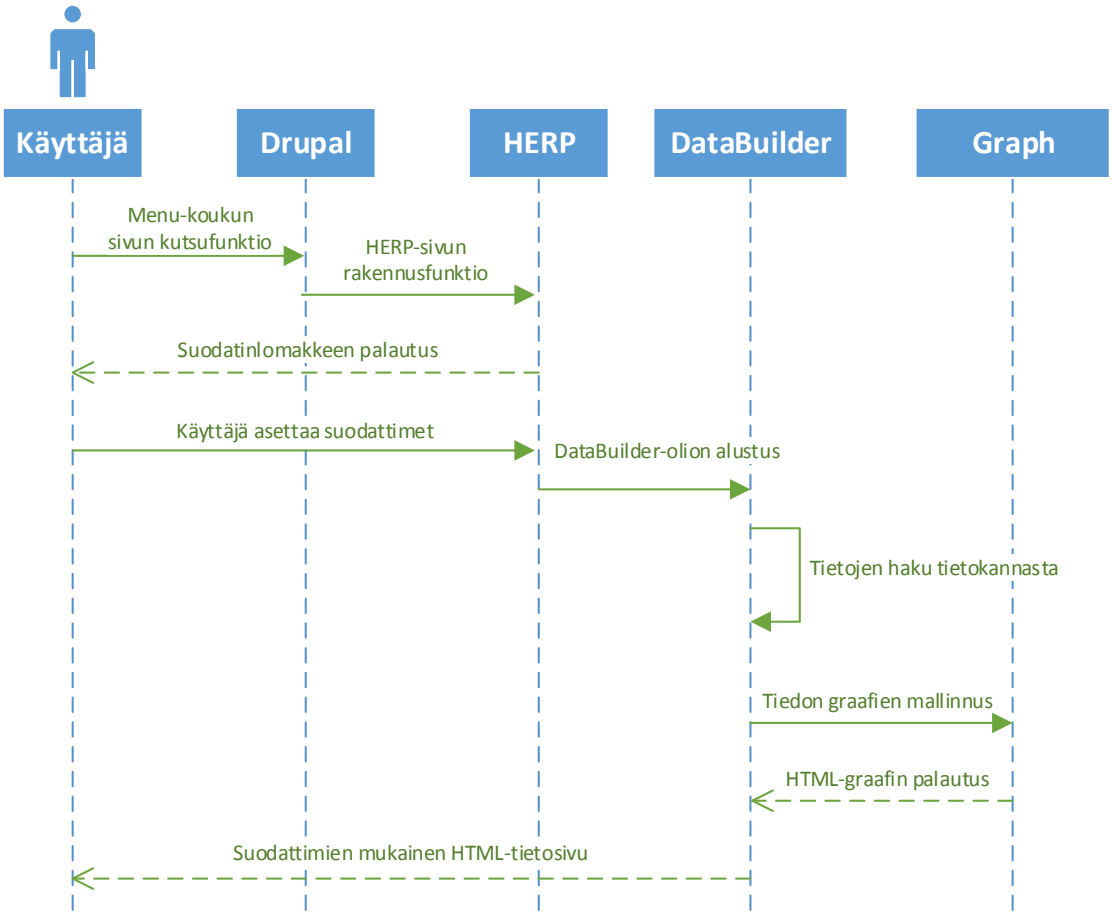
Graph.class.php on Graph-luokan tiedosto, joka hoitaa moduulien käyttämän grafiikan luomisen ja näyttämisen. Graph-olio alustaa grafiikan annettujen parametrien pohjalta ja palauttaa valmiin HTML- ja JavaScript-koodin.

Moduulikansiossa on myös moduulin käyttämiä kirjastoja, kuten *Chart.js* ja *Select2*. *Chart.js* tarjoaa käyttöön JavaScript-pohjaiset kaaviot. *Select2* puolestaan muuntaa tavalliset selaimissa esiintyvät valintaluettelot käyttäjäystävällisemmiksi ja mahdollistaa monivalintojen tekemisen vaivattomasti, kuten kuvassa 9 on esimerkissä osavaltiot sisältävässä valintaluettelossa. Käyttöliittymää ja käyttökokemusta parantavien kirjastojen käytöstä on suurta apua, koska kaikkea ei silloin tarvitse toteuttaa itse ja käyttäjän on helpompi omaksua käyttöliittymä.



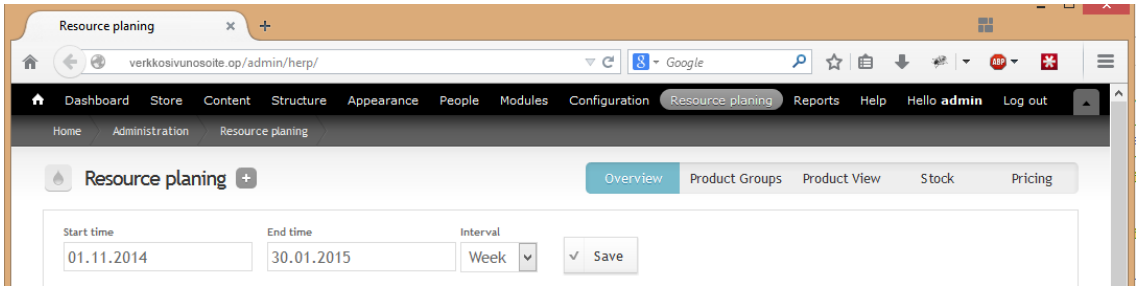
Kuva 9. Select2-alasvetovalikon esimerkki, jossa on valittuina kohteina Hawaii ja California.

Raportointisivuissa kaikki raporttityypit jakavat samanlaisen rakenteen ja toimintalogiikan, joissa ainoastaan raportin sisältö ja raportin suodattimet eriävät. Kuvassa 10 on sekvenssikaaviossa mallinnettu tyypillisen raportointisivun luontivaiheet ja luomiseen liittyvät osat. Drupal välittää käyttäjän sivupyynnöt HERP-moduulille, joka palauttaa käyttäjälle suodatinlomakkeen ja asetettujen suodattimien mukaisen raporttisivun. Suodattimien asettaminen on pakollista raportin luomiseksi, koska kaikkea tietoa ei ole tarpeellista tai edes järkevää näyttää kerralla.



Kuva 10. Raportointisivun luontivaiheet sivulatauksen aikana.

Yksi tärkeä moduulin käyttämä koukku on *hook_menu*, jolla päämoduuli esittelee käyttämänsä URL-polut Drupalille. Koodiesimerkissä 1 ovat ydinmoduulin esittelemien polkujen rakenteet, jotka Drupal tulkitsee omaan valikkojärjestelmään. Kuvassa 11 näkyy, miltä valikko näyttää sivustolla. Kuvassa mustapohjaisena on sivuston ylläpitäjän valikko, jossa ”Resource planning” -valikkoelementti on aktiivisena. Valkoisella sivupohjalla oikeassa reunassa ovat välilehdet ”Overview”, ”Product Groups” ja ”Product View”, jotka myös esitellään koodiesimerkissä 1.



Kuva 11. Selaimen näkymä, jossa moduulin valikkorakenne on esillä.

Koodiesimerkissä 1 välilehtien yhtenä attribuuttina on ”page callback”, joka kertoo Drupal-valikkojärjestelmälle, mitä funktiota kutsutaan, kun selaimella pyydetään kyseistä URL-osoitetta. Toinen attribuutti ”file” kertoo, mistä tiedostosta kyseinen kutsufunktio löytyy, jolloin Drupal automaattisesti sisällyttää tiedoston mukaan suoritukseen. Koodiesimerkissä kutsutaan HERP-moduulin raportointisivujen rakennusfunktioita, joissa varsinainen HTML-tulosteen rakentaminen suoritetaan.

```php

function herp_menu() {
  $items = array();

  $items['admin/herp'] = array(
    'title' => 'Resource planing',
    'page callback' => 'herp_overview_page',
    'access callback' => 'user_access',
    'access arguments' => array('administer products'),
    'description' => 'Administer eCommerce',
    'file' => 'herp.pages.inc',
  );

  $items['admin/herp/overview'] = array(
    'title' => 'Overview',
    'type' => MENU_DEFAULT_LOCAL_TASK,
    'weight' => -50,
  );

  $items['admin/herp/groups'] = array(
    'title' => 'Product Groups',
    'page callback' => 'herp_groups_page',
    'access callback' => 'user_access',
    'access arguments' => array('administer products'),
    'description' => 'Administer products',
    'file' => 'herp.pages.inc',
    'type' => MENU_LOCAL_TASK,
    'weight' => -30,
  );

  $items['admin/herp/products'] = array(
    'title' => 'Product View',
    'page callback' => 'herp_products_page',
    'access callback' => 'user_access',
    'access arguments' => array('administer products'),
    'description' => 'Administer products',
    'file' => 'herp.pages.inc',
    'type' => MENU_LOCAL_TASK,
    'weight' => -30,
  );

    return $items;
}

```

Koodiesimerkki 1. HERP-moduulin URL-rakenne ja valikkoelementit.

Koodiesimerkissä 2 on ”Overview”-sivun rakentava koodi, jonka valikkojärjestelmä kutsuu käyttäjän tullessa raportoinnin pääsivulle.

```php
function herp_overview_page() {
  herp_includes(); // Call module includes
  $out = "";

  $return = "";
  if($_SESSION['herp']['start_date'] &&
    $_SESSION['herp']['end_date'] &&
    $_SESSION['herp']['interval'])
  {
    $dataBuilder = new DataBuilder($_SESSION['herp']['start_date'],
                                   $_SESSION['herp']['end_date'],
                                   array('completed'),
                                   $_SESSION['herp']['interval']);
    $return = $dataBuilder->overview();
  }

  $form_vars = array('type' => 'overview');
  $out .= drupal_render(drupal_get_form('herp_filter_form', $form_vars));
  $out .= $return;

  return $out;
}
```

Koodiesimerkki 2. Overview-sivun rakentava funktio.

Sivulla tarkistetaan käyttäjän istunnosta, onko aikaväli ja aikajakso asetettu suodatinlomakkeessa. Kun ne ovat asetettuna, lähetetään arvot DataBuilder-oliolle, joka palauttaa valmiiksi muotoillun HTML-rakenteen päänäkymästä sisältäen kuvan 2 (s. 9) malliset tiedot ja kaaviot. Koodiesimerkissä 2 tapahtuu myös suodatinlomakkeen kutsuminen ja mallintaminen HTML-koodiksi *drupal_render*-funktion avulla. Suodatinlomaketta ladattaessa lähtee parametrina määritys, minkätyyppinen suodatinlomake palautetaan. Lomakkeen rakenne muuttuu raportointisivukohtaisesti, koska kaikkia lomaketietoja ei tarvita kaikilla sivuilla.

DataBuilder-osuus raportointisivuista

Kaikilla raportointisivuilla alustetaan DataBuilder-olio vaadituilla parametreilla, ennen kuin tietoja voi hakea. Koodiesimerkissä 3 on koodi, jossa oliolle annetut parametrit käsitellään ja tallennetaan olion muuttujiin. Annetuista parametreista aloitus- ja päätöspäivämäärä on annettu muodossa ”22.1.2015”, jolloin päivämäärä tulee ensin muuttaa Unix-aikaleimaksi suorittamalla *strtotime*-funktio päivämäärälle. Seuraavaksi olio tekee aloitus- ja päätösajasta uudet aikaleimat käyttäen *mktime*-funktiota. Uuden aikaleiman tarkoitus on siirtää ajankohdat päivän alku- ja loppusekunteihin. Toimenpide on oleellinen, jotta tarkasteluaikavälillä päivät ovat kokonaisuudessaan mukana.

```
18  function __construct($start, $end, $order_statuses, $target = 'month') {
19
20    $start = strtotime($start); // Convert back to timestamp
21    $end = strtotime($end);
22
23    // Modify date to be very start
24    $start_time = mktime(00, 00, 00, date('m', $start),  date('d', $start),  date('Y', $start));
25    // Same but for the very end of day
26    $end_time = mktime(23, 59, 59, date('m', $end),  date('d', $end),  date('Y', $end));
27
28    if(!$start_time || !$end_time) return false; // mktime failed
29
30    $this->target = $target;
31    $this->start = $start_time;
32    $this->end = $end_time;
33    $this->order_statuses = $order_statuses;
34
35  }
```

Koodiesimerkki 3. DataBuilder-olion alustusfunktio.

Olion alustuksessa myös määritellään tarkasteltavien tilauksien tila ja se, minkälaisella intervallilla tietoja koostetaan annetulla aikavälillä. Oletuksena käsiteltävien tilauksien tila on valmis, mutta ylläpitäjä voi haluta tarkastella peruuntuneiden tai palautettujen tilauksien tietoja.

”Overview”-raportointisivulla kutsutaan DataBuilder-olion alustuksen jälkeen *overview*-funktiota, joka rakentaa raporttisivun. Funktiossa ensimmäinen tehtävä on hakea tietokannasta kaikki olion alustuksessa annettujen parametrien mukaiset tilaukset, kuten koodiesimerkissä 4 on tehty.

```
445  /**
446   * Build overview page.
447   *
448   * @return themed output
449   */
450  public function overview() {
451    $out = "";
452
453    $data = $this->queryTotalSales();
454
455    $stockValue = $this->queryCurrentStockValue();
456
```

Koodiesimerkki 4. Overview-funktion tietojen haku.

Koodiesimerkin 4 *queryTotalSales*-funktio tekee tietokantaan haun ja palauttaa assosiatiivisen taulukon. Taulukko on osioitu aloituspäivästä loppupäivään aikajakson pituisina segmentteinä. Koodiesimerkissä 5 funktio pilkkoo annetun aikavälin haluttuihin aikajaksoihin.

```
737   /**
738    * Create time segments from start and end time.
739    *
740    * @return array of time segments with start and end.
741    */
742   private function getTimeSegments() {
743     $segments = array();
744
745     $start = $this->start;
746     while($start < $this->end) {
747       $end = strtotime('+1 '. $this->target, $start) -1; // Get time to next segment -1 second
748       $segments[] = array(
749         'start' => $start,
750         'end' => min($end, $this->end),
751       );
752
753       $start = $end +1;
754     }
755
756
757     return $segments;
758   }
```

Koodiesimerkki 5. Aikavälin pilkkominen aikajakson pituisiin segmentteihin.

Tietokannan tilaushaku tehdään jokaiselle aikasegmentille, ja tulokset koostetaan assosiatiiviseksi taulukoksi. Tietokannan SQL-lausekkeen kyselyn muodostamiseen käytetään Drupalin rajapinnasta löytyvää *db_select*-funktion palauttamaa SelectQuery-oliota. Olio rakentaa dynaamisesti SQL-lausekkeen annettujen parametrien pohjalta. Olion etuna ovat helpommin hallittavat monimutkaiset kyselyjen rakenteet, jossa kyselyn rakenne voi muuttua parametrien muuttuessa. Koodiesimerkissä 6 on tehty segmenttikohtainen tilaustietojen haku kahdella SQL-kyselyllä. Kyselyssä käytetään summafunktiota hakemaan segmentin aikavälillä olevien tilausten yhteenlaskettuja määriä erilaisista tilaustiedoista, kuten verojen määrä tai tilattujen tuotteiden sisäänostohinta.

Kyselyn tauluja joutuu liittämään JOIN-käskyillä useasta taulusta tietokannan normalisoinnin vuoksi. Pääasiallisena tietokantatauluna on kuitenkin uc_orders, josta selviää tilausnumero ja luontiaika. Tilausnumeron perusteella voi muut tarvittavat taulut liittää kyselyyn.

Koodiesimerkin 6 QuerySelect-oliot koostavat SQL-kyselyn annettujen funktiokutsujen ja parametrien pohjalta. Olio hoitaa automaattisesti syötettyjen tietojen sanitoinnin ja muodostaa lopuksi tietokantaohjelmiston mukaisen SQL-kyselyn.

```
604   foreach ($segments as $segment) {
605
606     // Get totals of sales, shipings, coupon usage
607     $query = db_select('uc_orders', 'uo');
608     $query->addExpression('COUNT(DISTINCT uo.order_id)', 'order_count');
609     $query->addExpression('SUM(ol.amount)', 'shipping_cost');
610     $query->addExpression('SUM(ol2.amount)', 'total_tax');
611     $query->addExpression('SUM(ol3.amount)', 'total_coupon');
612     $query->addExpression('SUM(uo.order_total)', 'order_total');
613     $query->leftJoin('uc_order_line_items', 'ol', 'uo.order_id = ol.order_id AND ol.type = :type',
614                      array(':type' => 'shipping')
615                     );
616     $query->leftJoin('uc_order_line_items', 'ol2', 'uo.order_id = ol2.order_id AND ol2.type = :type2',
617                      array(':type2' => 'tax')
618                     );
619     $query->leftJoin('uc_order_line_items', 'ol3', 'uo.order_id = ol3.order_id AND ol3.type = :type3',
620                      array(':type3' => 'coupon')
621                     );
622     $query->condition('uo.created', $segment['start'], '>=');
623     $query->condition('uo.created', $segment['end'], '<=');
624     $query->condition('uo.order_status', $this->order_statuses, 'IN');
625
626     $result = $query->execute()->fetchObject();
627
628     // Get totals of quantities, sell price, cost
629     $query2 = db_select('uc_order_products', 'up');
630     $query2->addExpression('SUM(qty)', 'total_qty');
631     $query2->addExpression('SUM(up.price * up.qty)', 'total_price');
632     $query2->addExpression('SUM(up.cost * up.qty)', 'total_cost');
633     $query2->leftJoin('uc_orders', 'uo', 'up.order_id = uo.order_id');
634     $query2->condition('uo.created', $segment['start'], '>=');
635     $query2->condition('uo.created', $segment['end'], '<=');
636     $query2->condition('uo.order_status', $this->order_statuses, 'IN');
637
638     $result2 = $query2->execute()->fetchObject();
```

Koodiesimerkki 6. Tilaustietojen kysely käyttäen kahta QuerySelect-oliota.

Koodiesimerkissä 7 on esimerkki yhden segmentin tietokantakyselyistä ennen paikkamerkkien korvaamista varsinaisilla tiedoilla. Paikkamerkkien käyttö on tietoturvallista ja estää SQL-injektiohyökkäyksiä.

```
1   SELECT COUNT(DISTINCT uo.order_id) AS order_count,
2   SUM(ol.amount) AS shipping_cost,
3   SUM(ol2.amount) AS total_tax,
4   SUM(ol3.amount) AS total_coupon,
5   SUM(uo.order_total) AS order_total
6   FROM
7   {uc_orders} uo
8   LEFT OUTER JOIN {uc_order_line_items} ol ON uo.order_id = ol.order_id AND ol.type = :type
9   LEFT OUTER JOIN {uc_order_line_items} ol2 ON uo.order_id = ol2.order_id AND ol2.type = :type2
10  LEFT OUTER JOIN {uc_order_line_items} ol3 ON uo.order_id = ol3.order_id AND ol3.type = :type3
11  WHERE  (uo.created >= :db_condition_placeholder_0) AND
12  (uo.created <= :db_condition_placeholder_1) AND
13  (uo.order_status IN  (:db_condition_placeholder_2))
14
15  SELECT
16  SUM(qty) AS total_qty,
17  SUM(up.price * up.qty) AS total_price,
18  SUM(up.cost * up.qty) AS total_cost
19  FROM
20  {uc_order_products} up
21  LEFT OUTER JOIN {uc_orders} uo ON up.order_id = uo.order_id
22  WHERE  (uo.created >= :db_condition_placeholder_0) AND
23  (uo.created <= :db_condition_placeholder_1) AND
24  (uo.order_status IN  (:db_condition_placeholder_2))
```

Koodiesimerkki 7. QuerySelect-olion tuottama SQL-kysely.

Raporttisivusta riippuen kyselyn rakenne ja palautettujen tietojen käsittelytapa eroavat, kuitenkin noudattaen samaa logiikkaa. ”Overview”-raporttisivulla haetuista tiedoista koostetaan yhteenvedot ja Graph-olion vaatimat tiedot viivagraafin luontiin. Graph-olio alustetaan antaen graafissa käytettävät tiedot ja graafin määrittelyt, kuten koodiesimerkissä 8 on tehty. Graph-olion graph-funktio palauttaa alustuksessa annettujen tietojen pohjalta valmiin HTML- ja JavaScript-koodin, joka liitetään suoraan sivun tulosteeseen.

```
585     $graph_values['title'] = 'Gross revenue';
586     $graph = new Graph(
587       'totalSales', 'line',
588       array(
589         'data' => array($graph_values),
590         'labels' => $labels),
591         array(
592           'responsive' => true,
593           'currency' => true,
594           'height' => 200,
595           'width' => 800
596         )
597     );
598
599     $out .= $graph->graph();
```

Koodiesimerkki 8. Graph-olion alustus ja viivagraafin tulostus.

Graph-olio toimii Chart.js-graafikirjaston alustusfunktiona, joka valmistelee parametrein määritellyn kaavion annetuilla tiedoilla. Graafikirjaston jokainen kaavio käyttää hieman erilaista alustusta ja tiedon esitysmuotoa, joten Graph-olio muotoilee tiedon syöttömuodon yhdenmukaiseksi riippumatta halutusta kaaviosta. Olio toteuttaa kirjaston yleisimmin käytettävät kaaviotyypit, jotka ovat pylväs-, viiva-, piirakka- ja hämähäkkikaaviot.

4.2 Raporttisivujen tunnuslukujen laskenta

ABCD-luokitus

Tuotteiden ja tuoteryhmien luokituksen laskenta tehdään DataBuilder-olion sisällä. Luokittelun laskenta alkaa laskemalla jokaisen luokiteltavan kohdan prosentuaalinen osuus kokonaistuotosta. Prosentuaalisen osuuden saa selville jakamalla kohdan tuoton koko-

naistuotolla. Luokittelun laskemiseksi tulee kohdat järjestää laskevaan järjestykseen prosenttiosuuden mukaan, jolloin listauksen iterointi alkaa merkittävimmistä kohdista. Koodiesimerkissä 9 on luokittelun suorittavan funktion rakenne.

```
443   /**
444    * Calculate ABC analysis out of given key value pair.
445    * @return Array of key and value as class
446    */
447   private function calculateABC($data) {
448     arsort($data, SORT_NUMERIC); // Sort data from highest to lowest
449     // Loop through sorted values and perform additions to find product classes A, B, C.
450     $data_total = 0;
451     $class = 'A'; // Starting group
452     $data_class = array();
453
454     if(!empty($data) && is_array($data)) foreach ($data as $key => $value) {
455       $data_total += $value;
456
457       if($value > 0 ) $data_class[$key] = $class;
458       else $data_class[$key] = 'D';
459
460       // Class A = 70% of data
461       if($data_total >= 0.7 && $class == 'A') {
462         $class = 'B';
463         $data_total = 0;
464       }
465
466       // Class B = 20% of data and C for rest
467       else if($data_total >= 0.2 && $class == 'B') {
468         $class = 'C';
469         $data_total = 0;
470       }
471
472       // Special D class for data where performance is 0 and under
473       else if($value <= 0 && ($class == 'C' || $class == 'B')) {
474         $class = 'D';
475         $data_total = 0;
476       }
477
478     }
479
480     return $data_class;
481
482   }
```

Koodiesimerkki 9. ABCD-luokittelun luokittelufunktio.

Luokittelufunktio lajittelee annetun taulukon arvot suuruusjärjestykseen laskevasti ja käy läpi taulukon arvot ja antaa luokittelun jokaiselle kohdalle alkaen A-luokasta. Iteroinnin yhteydessä jokaisen kohdan arvo summataan ja tarkastellaan, milloin saavutetaan luokkien rajat. A-luokan 70 % täyttyy ensin, sitten B-luokan 20 % ja loput C-luokkaan. Kohdat, joiden arvo on 0 tai sen alle, saa D-luokan. Taulukossa 2 on esimerkkinä eräiden avainarvojen prosenttiosuuksien pohjalta lasketut luokat käyttäen koodiesimerkin 9 funktiota.

Taulukko 2. ABCD-luokittelu annettujen arvojen perusteella.

Avainarvo	Prosenttiosuus	Luokka
103	0,38609506302521	A
62	0,35497636554622	A
2	0,10602678571429	B
105	0,095063025210084	B
91	0,028492647058824	C
61	0,016018907563025	C
5	0,013327205882353	C

Varastonkierron laskenta

Varastonkierto lasketaan jokaiselle valitun tuoteryhmän tuotteelle tuotetason listauksessa. Prosessi alkaa kyselyllä *herp_stock*-moduulin varastohistoriaan, josta haetaan annetun aikavälin varastokehitys listattaville tuotteille. Varastohistoria palautuu jaettuna aikaväleihin, joissa varastossa on tapahtunut muutoksia, kuten myyntitapahtuma tai tuotteen varastomäärän kasvatus. Järjestelmä laskee tuotteille varastonkiertonopeuden ja saatua nopeuslukua käyttäen varastonkiertonopeuden päivinä. Koodiesimerkissä 10 on järjestelmän käyttämä koodi tuotteiden varastonkierron laskentaan.

```
295   if(module_exists('herp_stock')) {
296     // Load histories for all product IDs with given start and end time
297     $histories = herp_load_history(null, array_keys($stock_turnover), $this->start, $this->end);
298
299     // Parse returned histories
300     $stock_value = array();
301     foreach ($histories as $model => $segments) {
302       $stock_value[$model] = 0;
303       foreach ($segments as $time => $vals) {
304         $stock_value[$model] += $vals['cost'] * $vals['stock'];
305       }
306     }
307
308     // Calculate turnover for each product
309     foreach ($stock_value as $model => $value) {
310       // Duration of days between start and end
311       $duration = round(($this->end - $this->start)/(60*60*24));
312       $turnover_ratio = 0; // Default turnover is 0
313       $divider = $value / count($histories[$model]);
314
315       if($divider > 0) { // 0 = no activity
316         $turnover_ratio = $stock_turnover[$model]['cost'] / $divider;
317         $rows[$model]['data']['it']['data'] = $this->numf($turnover_ratio, 2);
318         // Add calculated inventory turnover days to table
319         if($turnover_ratio > 0)
320           $rows[$model]['data']['it_days']['data'] = $this->numf($duration / $turnover_ratio , 0);
321       } else {
322         // Add no activity to products table.
323         $rows[$model]['data']['it']['data'] = "-";
324         $rows[$model]['data']['it_days']['data'] = "-";
325       }
326     }
327     // Add new table headers for turnover and turnover days. Abbr tag for readability.
328     $header[] = array('data' => t('<abbr title="Inventory Turnover">IT</abbr>'));
329     $header[] = array('data' => t('<abbr title="Inventory Turnover for ' .$duration.
330       ' days">IT ' .$duration. ' days</abbr>'));
331   }
```

Koodiesimerkki 10. Varastonkierron laskentalogiikka.

Varastonkierto lasketaan kaikille tuotteille riippumatta varastotilanteen muutoksista. Tuotteet, joiden varastossa ei ole muutoksia tapahtunut, saavat nopeuden arvon 0. Varastonkiertonopeutta hyväksi käyttäen saa selville, mikä olisi varaston myyntiaika päivinä, jos myyntinopeus olisi annettua aikaväliä vastaava.

Seuraavassa esimerkki erään tuotteen 365 päivän varastonkiertonopeuden laskennasta. Koko vuoden varaston keskiarvoksi tuotteelle on saatu hinta 203,254 €. Myytyjen tuotteiden hankintahinnaksi muodostuu 988,95 €.

$$4{,}865 = \frac{988{,}950\ €}{203{,}254\ €} \qquad (4)$$

Kun käytetään varastonkierron nopeuden laskentakaavaa, tulee tuotteelle koko vuoden varastonkiertonopeudeksi 4,865, jonka avulla saa selville, että tuotteen varasto kiertää vuodessa lähes viisi kertaa. Toisin sanoen tuotteen varasto myydään 75 päivän aikana vuodessa, kun käytetään FIFO-menetelmää varastoinnissa.

4.3 Varastohistorian toteutus

Varastohistoriasta vastaa HERP Stock -moduuli, joka luo merkinnät tietokantaan tuotteiden varaston muuttuessa. Moduuli myös rakentaa historian takautuvasti tilaustietojen perusteella. Tuotteiden varastojen täydentämiseksi tehtiin järjestelmään ostojen seurantaan lomake, jonka avulla tilauksien saavuttua oli mahdollista päivittää tuotteen varastomäärä.

Moduulin tärkein toiminto on varaston jatkuva seuranta ja muutoksien kirjoittaminen tietokantalokiin. Seurantaan käytetään Drupal-järjestelmän tarjoamia koukkuja, jolloin HERP Stock- ja Übercart stock -moduulit voivat keskustella keskenään ja ilmoittaa toisilleen, jos varastoon tulee muutos. Koodiesimerkissä 11 HERP Stock -moduuli kiinnittyy Übercart-moduulin lomakkeeseen, jolla muokataan varastoa.

```
798  /**
799   * Implements hook_form_FORM_ID_alter() for uc_stock_edit_form().
800   */
801  function herp_stock_form_uc_stock_edit_form_alter(&$form, &$form_state, $form_id) {
802    array_unshift($form['#submit'], 'herp_stock_uc_stock_admin_edit');
803  }
804
805  function herp_stock_uc_stock_admin_edit($form, &$form_state) {
806
807    foreach (element_children($form_state['values']['stock']) as $id) {
808      $stock = $form_state['values']['stock'][$id];
809
810    $current_data = db_query('SELECT p.cost, n.nid, s.stock
811  from uc_products as p
812  inner join node as n ON p.nid = n.nid AND p.vid = n.vid
813  left join uc_product_stock as s ON p.nid = s.nid
814  where model = :model', array(':model' => $stock['sku']))->fetchObject();
815
816      $values = array(
817        'nid' => $current_data->nid,
818        'model' => $stock['sku'],
819        'cost' => $current_data->cost,
820        'stock' => (int) $stock['stock'],
821        'old_stock' => (isset($current_data->stock)) ? $current_data->stock : 0,
822        'time' => time(),
823      );
824
825      herp_stock_insert_history($values);
826      watchdog('herp_stock', ':model stock adjusted from :stock to :new_value', array(':model' => $
           stock['sku'], ':stock' => $current_data->stock, ':new_value' => $stock['stock']),
           WATCHDOG_NOTICE);
827
828      dpm($stock);
829    }
830  }
```

Koodiesimerkki 11. HERP Stock -moduulin käyttämä koodi varastonseurantaan.

herp_stock_form_uc_stock_edit_form_alter-funktio on koukku, jolla pääsee muokkaamaan järjestelmän muita lomakkeita. Koodiesimerkissä rivillä 802 lomakkeeseen lisä-

tään oma kutsufunktio, joka tekee varastonmuutoksesta lokimerkinnän raportoinnin tietokantaan. Tällä koodilla kauppiaan tekemät tuotteen varastomuutokset tulevat talteen. Koodi ei kuitenkaan osaa ottaa kiinni tilaustapahtumaa, koska siihen tarvitaan eri koukku.

Tilauksen yhteydessä tapahtuvan varastomuutoksen saaminen kiinni vaatii oman koukun. Koodiesimerkissä 12 on koodi, jolla verkkokaupan varastonmuutoksen saa kiinni varastomoduulissa. *herp_stock_uc_stock_adjusted*-funktio tulee kutsutuksi Übercart-moduulien toimesta aina, kun varastossa tapahtuu muutos. Tällainen muutos on esimerkiksi tilaus. Funktiota ei kuitenkaan kutsuta, kun yrittäjä tekee suoraan tuotteeseen varastomuutoksen, ja siten järjestelmässä tarvitaan varastonseurantaan kaksi koukkua.

```
760 /**
761  * Implements hook_uc_stock_adjusted.
762  *
763  * Stock is changed for item.
764  *
765  * @param $model
766  *   The SKU whose stock level is being changed.
767  * @param $stock
768  *   The stock level before the adjustment.
769  * @param $qty
770  *   The amount by which the stock level was changed.
771  */
772 function herp_stock_uc_stock_adjusted($model, $stock, $qty) {
773   $new_value = $qty+$stock;
774
775   $current_data = db_query('SELECT p.cost, n.nid, s.stock
776 from uc_products as p
777 inner join node as n ON p.nid = n.nid AND p.vid = n.vid
778 left join uc_product_stock as s ON p.nid = s.nid
779 where model = :model', array(':model' => $model))->fetchObject();
780
781   if(empty($current_data)) {
782     watchdog('herp_stock', 'FAILED :model stock adjusted from :stock to :new_value. Invalid value
        given.', array(':model' => $model, ':stock' => $stock, ':new_value' => $new_value),
        WATCHDOG_ERROR);
783   } else {
784     $values = array(
785       'nid' => $current_data->nid,
786       'model' => $model,
787       'cost' => $current_data->cost,
788       'stock' => $new_value,
789       'old_stock' => $stock,
790       'time' => time(),
791       );
792     herp_stock_insert_history($values);
793     watchdog('herp_stock', ':model stock adjusted from :stock to :new_value', array(':model' => $
        model, ':stock' => $stock, ':new_value' => $new_value), WATCHDOG_NOTICE);
794   }
795
796 }
```

Koodiesimerkki 12. Moduulin esittämä koukku varastomuutoksien seurantaan.

HERP Stock -moduulin toinen tärkeä toiminto on kyky rakentaa varastohistoria pohjautuen tilauksien varastomuutoskommentteihin ja olettamuksiin myymättömien tuotteiden

varastotilanteista tiettyinä ajanhetkinä. Historian rakentaminen vaatii kaikkien järjestelmän valmiiden tilauksien ja tilauksiin liittyvien tuotteiden hakemisen tietokannasta. Koodiesimerkissä 13 on koko historian rakennusprosessi tilauksiin pohjautuvan tiedon avulla. Koodissa ladataan tilausnumeron mukaan järjestetty listaus tilauksista ja tilauksen kommentit parsitaan rekursiivisesti läpi etsien varastomuutoksiin viittavia kommentteja.

```
1490  /**
1491   * Parse through orders and order comments to build stock history for all available
      products
1492   *
1493   *
1494   */
1495  function herp_build_stock_history() {
1496
1497    // Clear up table from data.
1498    $truncate = db_truncate('herp_stock_history')->execute();
1499
1500    // Get all completed orders ordered by order_id
1501    $query = db_select('uc_orders', 'o');
1502    $query->fields('o', array('order_id'));
1503    $query->condition('order_status', 'completed', '=');
1504    $query->orderBy('order_id', 'ASC');
1505    $result = $query->execute();
1506
1507    foreach ($result as $row) {
1508      $order = uc_order_load($row->order_id);
1509      $comments = uc_order_comments_load($order->order_id, true);
1510      $products = array();
1511      foreach ($order->products as $product) {
1512        if(empty($product->nid) && empty($product->model)) continue;
1513        $products[$product->nid]['nid'] = $product->nid;
1514        $products[$product->nid]['model'] = trim($product->model);
1515        $products[$product->nid]['cost'] = $product->cost;
1516        $products[$product->nid]['qty'] = $product->qty;
1517        $products[$product->nid]['time'] = $order->created;
1518      }
1519      foreach ($products as $nid => $product) {
1520        $stock_found = false;
1521        foreach ($comments as $comment) {
1522          if(!empty($product['model']) && stristr($comment->message, $product['model'])) {
1523            $stock_msg = stristr($comment->message, 'decreased to ');
1524            if($stock_msg !== FALSE) {
1525              $stock_found = true;
1526              $stock_amount = (int) str_replace('decreased to ', '', $stock_msg);
1527              $products[$product['nid']]['time'] = (!empty($comment->created)) ? $comment->
                created : $order->created;
1528              $products[$product['nid']]['old_stock'] = $products[$product['nid']]['qty'] +
                $stock_amount;
1529              $products[$product['nid']]['stock'] = $stock_amount;
1530            } else {
1531
1532            }
1533          }
1534        }
1535      }
1536
1537      foreach ($products as $nid => $product) {
1538        herp_stock_insert_history($product, false);
1539      }
```

Koodiesimerkki 13. Tilauksien käsittely ja historian luonti.

Koodiesimerkki parsii kaikki tilaukset ja tekee niiden pohjalta perustan varastohistorialle. Historiaa ei ole tarkoitus rakentaa useaan otteeseen alusta, koska nykyisessä mallissa olemassa oleva historia tyhjennetään *db_truncate*-funktiolla, jolloin historian tietokanta-taulu alustetaan uudelleen ja taulussa oleva tieto katoaa.

Koodiesimerkissä 14 myymättömät tuotteet haetaan tietokannasta *JOIN*-lausekkeilla ja *null*-ehtolausekkeella, jonka avulla saa listauksen kaikista näistä tuotteista.

```
1195  /**
1196   * Get list of unsold products and needed details for product to be inserted
      into history table.
1197   *
1198   */
1199  function herp_stock_unsolds() {
1200        $query = db_query('select DISTINCT
1201  sku as model,
1202  s.nid as stock_nid ,
1203  op.nid as orders_nid,
1204  s.stock as stock,
1205  n.created as time,
1206  p.cost
1207  from {uc_product_stock} as s
1208  left join {node} as n ON s.nid = n.nid
1209  left JOIN {uc_products} as p ON n.nid = p.nid AND n.vid = p.vid
1210  LEFT JOIN {uc_order_products} as op ON p.nid = op.nid
1211  where op.nid is null
1212  group by sku asc');
1213
1214
1215      $results = array();
1216      foreach($query as $row) {
1217        $results[$row->model] = array('stock' => $row->stock, 'cost' => $row->cost
            , 'nid' => $row->stock_nid, 'time' => $row->time);
1218      }
1219
1220      return $results;
1221  }
```

Koodiesimerkki 14. Tilaamattomien tuotteiden listaus.

Yhdistämällä ja luomalla varastonhistoriamerkinnät tilattujen tuotteiden ja tilaamattomien tuotteiden kohdalla on lähes koko varastohistoriakehitys saatu rakennettua. Järjestelmä ei kuitenkaan osaa ottaa kantaa tuotteisiin, jotka on mahdollisesti poistettu verkkokau-pasta ennen reaaliaikaisen varastohistorian seurannan käynnistämistä (HERP Stock -moduulin esittämät koukut). Tilauksista löytyy kuitenkin tieto kaikista ostetuista tuotteista, myös poistetuista tuotteista, mutta näiden tietojen parsiminen tuotti enemmän työtä kuin tuloksia.

Moduulin luoman varastohistoriataulun avulla on mahdollista selvittää, minkälainen varastotilanne on ollut tiettynä ajanhetkenä. Tätä historiatietoa käytetään varastonkierron laskennassa ja varastonkehitysraportissa. Koodiesimerkissä 15 on kuvattuna funktio, jolla eri moduulit voivat hakea tietoa tuotteen tai tuotteiden varastotilanteesta ajan parametrilla.

```
1043  /**
1044   * Load history of all avilable products at the time.
1045   * @time int - Timestamp of wanted history time
1046   * @value_only boolean - return only combined cost value
1047   */
1048  function herp_stock_history($time, $value_only = false) {
1049
1050    $query = db_query('Select h1.*
1051      FROM {herp_stock_history} h1
1052      INNER JOIN ( SELECT max(time) maxTime, model, stock FROM {herp_stock_history}
1053        WHERE time <= :time GROUP BY model
1054      ) h2
1055      ON h1.model = h2.model
1056      AND h1.time = h2.maxTime
1057      ORDER BY h1.model asc', array(':time' => $time), array('fetch' => PDO::FETCH_ASSOC));
1058
1059    $value = 0;
1060    $results = array();
1061    foreach($query as $row) {
1062      if($value_only) $value += $row['stock'] * $row['cost'];
1063      $results[$row['model']] = $row;
1064    }
1065    if($value_only) return $value;
1066
1067    return $results;
1068
1069  }
```

Koodiesimerkki 15. Historiahaun funktion rakenne.

Koodiesimerkin funktio muodostaa SQL-lausekkeen käyttäen sisähakua löytääkseen varastohistoriasta viimeisimmän varastomerkinnän tuotteelle, minkä jälkeen varastomerkinnät ryhmitellään tuotenumeron perusteella niin, että tuotteelle jätetään viimeisin historiamerkintä.

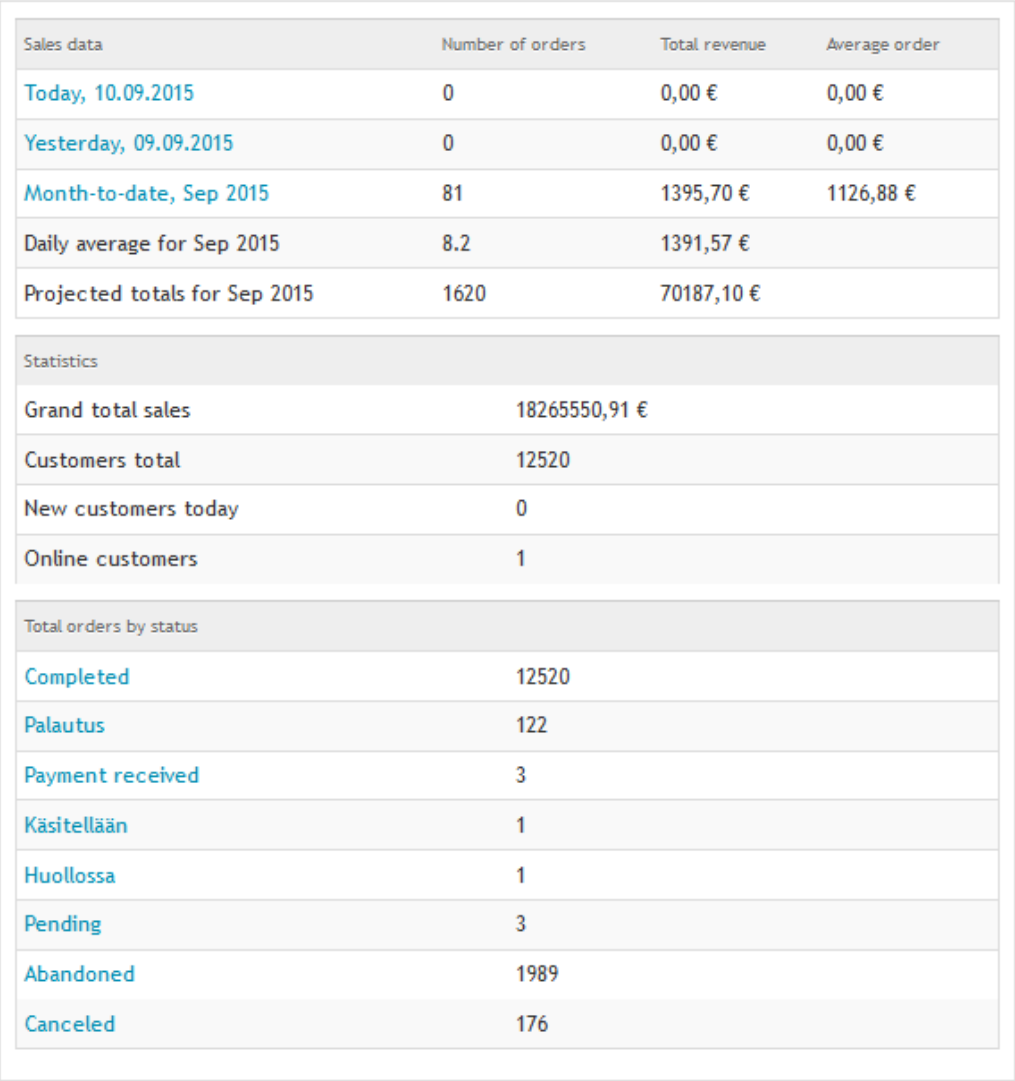
5 Testaus ja tulokset

5.1 Vertailu vastaavanlaisiin toteutuksiin

Insinöörityössä tehdyn järjestelmän vertailu vastaavanlaisiin osoittautui hankalaksi, koska kauppiaan raportteja visualisoivia järjestelmiä ei löytynyt ilmaisena tai kokeiluun soveltuvina versioina. Übercart sisältää oman raportoinnin, jota voi käyttää vertailukohteena, vaikka se ei sisällä tiedon visualisointia. Toinen mahdollinen vertailukohde on Google Analytics, joka sisältää verkkokauppoihin optimoidun tietolistauksen ja tiedon visualisoinnin.

Übercart-myyntiraportit

Übercart-raportit sisältävät yksinkertaiset listaukset myyntiraporteista. Niissä tiedot ovat vuosi-, kuukausi-, viikko- ja päivätasolla. Raportteihin on mahdollista lisätä tuotekohtainen myyntimäärä näkyviin, mutta niissä ei ole mahdollista rajata raporttia näyttämään tiettyä tuoteryhmää tai kategoriaa. Kuvassa 12 on myyntiraporttien pääsivu, jossa näytetään myös kuukauden myyntiennuste, joka on karkeasti laskettu myyntimääristä kuukauden alusta alkaen. Ennuste ei ota huomioon kausivaihtelua tai erikoispäiviä, jolloin ennuste on parhaimmillaan suuntaa antava. Übercart sisältää myös omat raportit varastonhallintaan, mutta siinä ainoana näkymänä on nykyhetken varastotilanne eikä varastonkehitystä ole siitä mahdollista saada selville.



Sales data	Number of orders	Total revenue	Average order
Today, 10.09.2015	0	0,00 €	0,00 €
Yesterday, 09.09.2015	0	0,00 €	0,00 €
Month-to-date, Sep 2015	81	1395,70 €	1126,88 €
Daily average for Sep 2015	8.2	1391,57 €	
Projected totals for Sep 2015	1620	70187,10 €	

Statistics	
Grand total sales	18265550,91 €
Customers total	12520
New customers today	0
Online customers	1

Total orders by status	
Completed	12520
Palautus	122
Payment received	3
Käsitellään	1
Huollossa	1
Pending	3
Abandoned	1989
Canceled	176

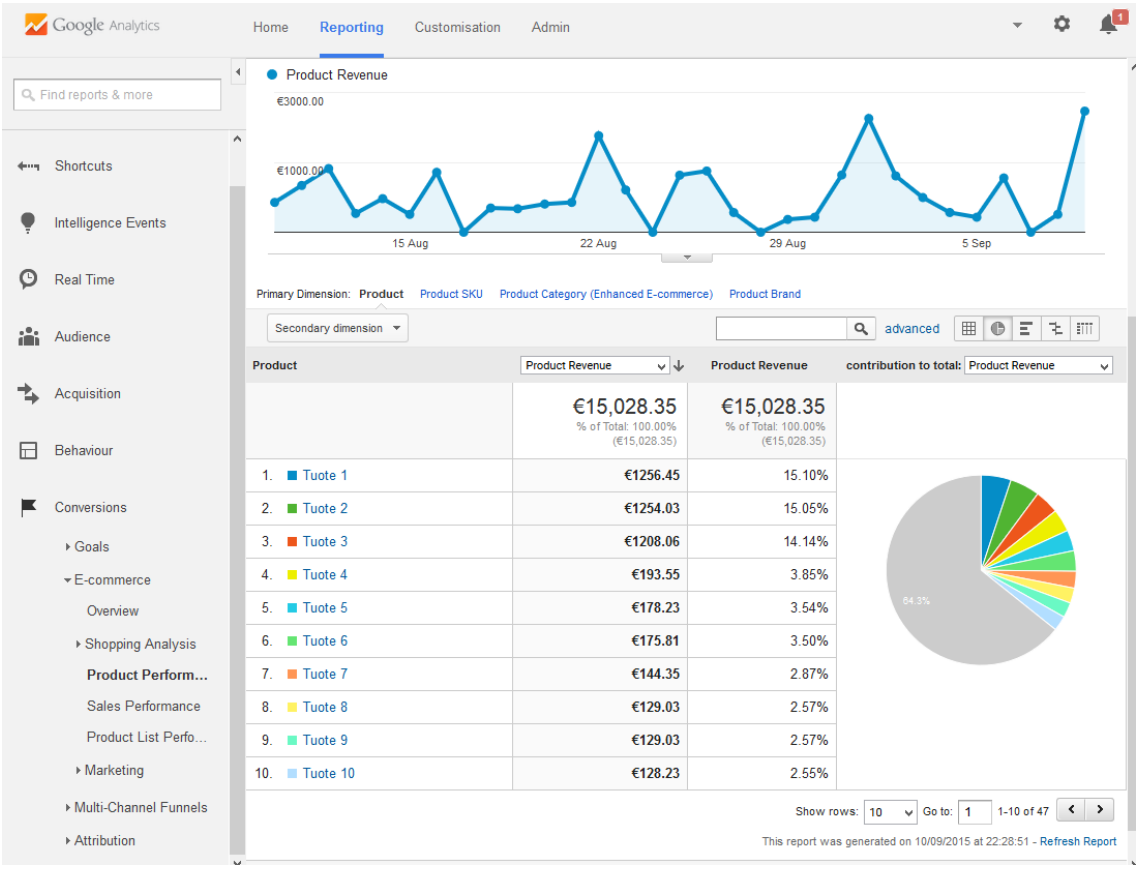
Kuva 12. Übercart-myyntiraporttien yleisnäkymä.

Raporteista käy hyvin ilmi lukujen kautta, minkälaisessa tilanteessa verkkokauppa on. Kauppiaalla voi kuitenkin olla vaikeuksia hahmottaa kehitystä pelkkien lukujen avulla, ja grafiikka olisi tällöin tärkeä elementti avustamaan hahmottamista. Toinen puute on, että raportit eivät kerro suoraan tuotteiden kehitystä eli huonosti kiertävät tuotteet eivät paljastu raporteista. Raporteissa on kuitenkin mahdollista koostaa lista kaikista myydyistä tuotteista halutulla aikavälillä.

Übercart-raportit ovat perusta, jota lähdettiin parantamaan ja avaamaan HERP-raportoinnin avulla. Übercart-raporteissa on riittävästi tietoa myynnillisesti, varastotilanteen tasolla ja yleistasolla, mutta nämä tiedot rajoittuvat pitkälti nykytilanteeseen eivätkä auta selvittämään varastokehitystä pitkällä ajanjaksolla tai anna visuaalista kuvaa myynnin kehittymisestä erilaisilla ajanjaksoilla.

Google Analytics -myyntiraportit

Google Analytics on Googlen kehittämä verkkosivujen kävijäseurantaan kehitetty palvelu. Analytics sisältää verkkokauppoihin räätälöidyn osion, jossa perinteisen analytiikan rinnalla on myös tilaustietoja, kuten ostetut tuotteet, tuotto, tuotteen myyntimäärä ja tilauksen summa. Kerätty tieto on mahdollista yhdistää muuhun sivustolta kerättyyn tietoon Analytics-järjestelmän sisällä. Kerättyjä tietoja on mahdollista tarkastella lukuina ja graafisella esitysmuodolla erilaisina kaavioina. Kuvassa 13 on esimerkki Google Analytics -palvelun myyntiraportista, jossa näytetään tuotekohtainen suoriutuminen.



Kuva 13. Google Analytics -palvelun myyntiraportti [17].

Tiedot kerätään Google Analytics -JavaScript-koodin lisäämän viestintäkanavan kautta. Tässä suurin ongelma on, että kaikki myyntitiedot eivät välity Analytics-palveluun ostotilanteissa. Käyttäjän selaimessa saattaa olla seurantaesto, tai huonolla yhteydellä Analytics-palvelin vastaa pyyntöihin liian hitaasti ja tämän takia myyntitilasto ei koskaan saavu analytiikkaan. Analytics tarjoaa esimerkillisen listauksen myyntiraportista, mutta tietojen oikeellisuuden puuttuessa ei annettuihin raportteihin voi varmasti luottaa.

Google Analytics tarvitsee verkkokaupan raporttien toimintaan sivuston lähdekoodiin upotettavan JavaScript-koodin lisäksi syvemmän integraation verkkokauppaan, jotta tilauksien tietoja pystyy lähettämään palveluun. [18.]

Google Analytics -raporttien tarjoama monipuolisuus ja muokattavuus ovat erittäin hyviä ja myyntiraportista paljastuu kattavasti tietoa. Raportin tietoja viestitään kaavioiden ja taulukkojen muodossa käyttäjälle, ja raportit ovat helppoja ymmärtää. Raportit eivät ole kauppiaalle kovin hyödyllisiä, koska niistä puuttuu osa myynneistä. Raportteja voi käyttää suuntaa antavana tietona.

5.2 Raporttien oikeellisuuden vahvistaminen

Raporttijärjestelmän raportteja testattiin kuvitteellisella tiedolla, samoin myös oikealla verkkokaupan tiedolla. Annettujen raporttien oikeellisuus tarkastetaan Übercart-raporttien kanssa, koska molemmat perustuvat samaan lähdetietoon ja toimivat rinnakkain samassa järjestelmässä.

Myyntiraportit

Myyntiraporttien oikeellisuuden laskenta toteutettiin tekemällä vertailua Übercart-raportteihin. Kaupan yleisnäkymässä oleva kokonaismyynti laskee oikein annetuilla aikaväleillä. Myynnistä piirtyvät kaaviot ovat myös oikeanlaisia; ne tarkistettiin manuaalisilla laskentatoimenpiteillä tarkistamalla sattumanvaraisesti valittujen aikajaksojen tilaukset.

Tuoteryhmien myyntitilastot olivat ongelmallisia tarkistaa, koska Übercart ei oletuksena ymmärrä käsitellä tuotteita tuoteryhmien kautta. Tarkistuslaskujen perusteella saadut raportit osoittautuivat pääasiallisesti oikeiksi. HERP-moduulien käyttöönoton jälkeiset tulokset tulevat virheettömästi, mutta sitä aiemmat raportit saattavat sisältää pienen virheen. Virhe selittyy tilanteella, jossa ylläpitäjä on poistanut kaupasta tuotteen piilottamisen sijaan. Tällöin tuotteen suhde tuoteryhmään on rikkoutunut eikä ole suoraan mahdollista riittää tuoteryhmän raporttiin. Ongelma sivutettiin etsimällä tuoteryhmän nimeä tuotteen nimestä, mutta tämä tapa ei ole täysin toimiva, joten virhemarginaali raporteissa on olemassa.

Tuotteiden myyntiraportit oli järjestelmässä helpompi tarkistaa, koska tiedot sai suoraan tilaustietokannasta. Tuotteiden myynti- ja varastosaldot tulevat suoraan tietokannasta, ja niiden tulkitaan siten olevan oikein. Tuotteen varastonkierron oikeellisuus laskettiin manuaalisesti hakemalla tietokannasta varaston keskiarvot ja myynnit. Varastonkierron laskenta annetulla aikavälillä siis osoittautui oikeaksi pois lukien aika ennen HERP-moduulin käyttöönottoa, koska paljon varastohistoriasta perustuu olettamuksiin ja tilauksista saatuihin tietoihin. Näissä tilanteissa varastonkierron voidaan sanoa olevan suuntaa antava, koska oikeellisuutta ei pystytä takaamaan.

ABC-analyysin tekeminen tuoteryhmille onnistui myös yhtä lailla kuin tuotteiden välinen vertailu. Tuotteiden samankaltaisuus auttaa tässä huomattavasti, koska myyntimäärät ja hinnat ovat verkkokaupassa samaa suuruusluokkaa.

Varastoraportit

Varastoraportin rakentaminen oli toteutuksessa hyvinkin haastavaa. Verkkokauppa on ollut käytössä usean vuoden ennen HERP-moduulin käyttöä, ja siksi varastohistorian rakentaminen on mahdollista vain tilauksien avulla. Tuotteen varastohistorian luonti onnistui suuntaa antavasti, mutta esimerkiksi varastotäydennyksien sijoittaminen ajankohtiin oli täysin arvaukseen perustuva. Moduulin käyttöönoton jälkeen kaikki varastomuutokset merkitään varastohistoriaan, jolloin kaikkien muutoksien ajankohdat tallentuvat oikein ja tarkasti.

Varastoraportin tuloksien oikeellisuus tarkistettiin ottamalla satunnaisia aikavälejä ennen moduulin käyttöönottoa ja sen jälkeen tehtiin tarkistuslaskelmia valituille tuotteille. Tuotteiden varastokehitystä verrattiin tilaustietokannan tilauskommentteihin varastovähennyksistä. Moduulin käyttöönoton jälkeiset merkinnät osoittautuivat tarkoiksi, ja tuotteen varastokehitys pystyttiin selvittämään. Ennen moduulin käyttöönottoa historiassa esiintyy aukkoja, jotka esimerkiksi eivät takaa varastonkiertoon saatua lukua oikeaksi. Saadut tulokset olivat kuitenkin suurusluokaltaan oikeanlaisia.

Täydellistä varastohistoriaa verkkokaupan alusta alkaen ei siis ole mahdollista tuottaa, ja tämän selvittäminen jäi myös pienemmälle painopisteelle työn priorisoinnissa.

6 Yhteenveto

Insinöörityössä luodun raportointityökalun käyttöönoton jälkeen liiketoiminnan kehitystä on seurattu raporttien avulla. Yrittäjä on niiden pohjalta pystynyt tekemään varmempia päätöksiä tulevien varastotäydennyksien suhteen. Lisäksi yrittäjällä on parempi kuva ja kontrolli tuotteista, jotka eivät liiku varastosta ulos riittävän nopeasti. Työssä saatiin tehtyä työkalut täysimääräiseen varaston valvontaan ja kattava informaatiolähde päätöksentekoa auttamaan.

Liitteessä 1 ovat yrittäjän kommentit toteutetusta järjestelmästä puolen vuoden käytön jälkeen. Kommenteissa varmistuu projektille asetettujen vaatimuksien täyttyminen ja tavoitteen saavuttaminen. Järjestelmän tuottamien tuloksien mittaaminen tavanomaisesta kasvusta ei ole onnistunut, mutta se ei ollut projektin pääasiallinen onnistumisen mittari. Varastoraporttien tuoman tiedon ansiosta tilausperusteet ovat muuttuneet yrityksessä, ja tilauksille on konkreettinen tieto tukemaan päätöstä.

Työn toteutuksessa esiintyi erilaisia ongelmia, joista suurimmat keskittyivät tiedon louhintaan raportteja varten. Raportteja varten tarvitaan suuria tietomääriä, ja paljouden vuoksi myös tietokantakyselyt saattoivat olla paikoittain hitaita ja monimutkaisia. Tietokantakyselyihin tehtiin jossain määrin optimointia, ja tietoa koostettiin mahdollisimman valmiiseen muotoon jo tietokantakyselyissä, jolloin PHP-puolella ei tarvitse suorittaa suuria laskentatehtäviä. MySQL-tietokannasta ei itsessään sisällä kovin laajalti analyyttisiä funktioita, joten esimerkiksi kumulatiivisen summan rakentaminen pelkästään tietokantakyselyllä ei ollut mahdollista. Kuitenkin kaikki ongelmalliset laskentatehtävät oli mahdollista suorittaa PHP-koodissa, joten aina löytyi ratkaisu.

Toinen ongelmakohta oli koodien dokumentointi. Paikoittain dokumentaatiota ei ollut tai se oli hyvin suppeaa. Tästä aiheutui ongelmia myöhemmin koodiin palatessa, koska tehdyt koodit eivät olleet enää helposti muistettavissa eikä niiden tarkoitusperä välttämättä auennut heti. Työn edetessä dokumentaatio parani ja koodin iteroinnin tuloksena syntyi selkeämmin kommentoitu ja jäsennelty rakenne. Hankalat ja monimutkaiset funktiot on kommentoitu selitteiden avulla, jotta myöhemmin funktioita katsoessa ymmärtäminen olisi helppoa.

Kolmantena asiana mainittakoon työn teknisyys. Paljon ajasta kului asioiden suunnitteluun ja opiskeluun oikeiden lähestymistapojen löytämiseksi. Drupal on varsin tuttu järjestelmä, mutta insinöörityössä useat käytetyistä asioista olivat uusia ja vaativat aikaa sisäistämiseen ja koodin testaamiseen. Useassa kohdassa työtä toteutustapa muuttui, kun löytyi järkevämpi tapa toteuttaa jokin asia. Työ vaati runsaasti Drupal-rajapinnan opiskelua ja vanhojen asioiden kertausta, mutta se on hyvä asia. Verkkokaupan kehitys ei lopu insinöörityön jälkeen, vaan saatua tietotaitoa käytetään edelleen jatkokehityksessä. Vastaavanlaista osaamista en itse pystyisi hankkimaan pelkästään kirjoja lukemalla.

Automaattinen hinnoittelu jäi työstä toteutuksen osalta pois, koska alkuperäinen toimintoja koskeva suunnitelma muuttui. Jatkokehityksenä hinnoittelu haluttiin koskemaan kaikkia kaupan tuotteita. Hinnoitteluun tarvitaan siis suhteutus myynnin kausiluonteiseen vaihteluun ja myyntiennusteiden luonti, joten toteutus rajattiin insinöörityöstä pois. Kuitenkin taustatyötä toimintojen eteen tehtiin, ja nykyisen järjestelmän ja uusien raporttitietojen myötä on mahdollista kehitellä myyntiennusteita pohjautuen tuotteen historiatietoihin.

Järjestelmä on perusta jatkosuunnitelmille, joiden avulla liiketoimintaa tehostetaan entisestään ja toimintoja automatisoidaan pidemmälle. Pitkällä tähtäimellä suunnitelmana on kasvattaa liiketoimintaa pitäen hallinnasta aiheutuvat lisätyöt kevyinä. Tähän tarvitaan edistyksellisiä ratkaisuja, joissa järjestelmä osaa tehdä päätöksiä automaattisesti ja yrittäjä asettaa vain parametrit.

Lähteet

1 Rama, Murthy. 2005. Production and Operations Management. Noida: New age international publishers.

2 Sakki, Jouni. 2014. Tilaus-toimitusketjun hallinta – Digitalisoitumisen haasteet. Vantaa: Jouni Sakki Oy.

3 Joseph Guran. 2009. Verkkodokumentti. <http://www.economist.com/node/13881008>. 19.6.2009. Luettu 15.6.2015.

4 Johnston, Jeff. 2014. The Pareto Principle (80-20 Rule). Verkkodokumentti. Investopedia. <http://www.investopedia.com/video/play/pareto-principle-8020-rule/>. 10.9.2014. Luettu 14.6.2015.

5 Fagerlund, Kirsi & Kärkkäinen, Sami. 2010. Yrityksen raaka-ainevarastonkierto. Opinnäytetyö. Laurea-ammattikorkeakoulu.

6 Sakki, Jouni. 2014. Vaihto-omaisuuden tunnusluvut, esimerkkinä päivittäiskauppa. Verkkodokumentti. <http://jounisakki.fi/blogi/?p=61>. 1.9.2014. Luettu 10.10.2015.

7 Inventory Turnover Ratio. 2014. Verkkodokumentti. My Accounting Course. <http://www.myaccountingcourse.com/financial-ratios/inventory-turnover-ratio>. Luettu 5.2.2015.

8 Wailgum, Thomas. 2007. ERP Definition and Solutions. Verkkodokumentti. <http://www.cio.com/article/2439502/enterprise-resource-planning/erp-definition-and-solutions.html>. 3.7.2007. Luettu 2.11.2015.

9 Honkanen, Simo & Virtanen, Seppo. 2012. Varastonhoitajan käsikirja. Tallinna: Sho Business Development.

10 System requirements. 2015. Verkkodokumentti. Drupal. <https://www.drupal.org/requirements>. Luettu 19.6.2015.

11 Usage statistics for Drupal core. 2015. Verkkodokumentti. Drupal. <https://www.drupal.org/project/usage/drupal>. 4.11.2015. Luettu 4.11.2015.

12 Glossary - Node. 2015. Verkkodokumentti. Drupal. <https://www.drupal.org/glossary#node>. Luettu 10.10.2015.

13 Glossary - Nid. 2015. Verkkodokumentti. Drupal. <https://www.drupal.org/glossary#nid>. Luettu 10.10.2015.

14 Glossary - Entity. 2015. Verkkodokumentti. Drupal. <https://www.drupal.org/glossary#entity>. Luettu 10.10.2015.

15 Understanding the hook system for Drupal modules. 2015. Verkkodokumentti. Drupal. <https://www.drupal.org/node/292>. Luettu 10.10.2015.

16 function hook_menu. 2015. Verkkodokumentti. Drupal <https://api.drupal.org/api/drupal/modules!system!system.api.php/function/hook_menu/7>. 19.10.2015. Luettu 19.10.2015.

17 Google Analytics. 2015. Verkkodokumentti. Google. <https://analytics.google.com/analytics/web/>. Luettu 10.9.2015.

18 Set up the web tracking code. 2015. Verkkodokumentti. Google. <https://support.google.com/analytics/answer/1008080>. Luettu 10.9.2015.

Yrittäjän kommentit toteutuneesta järjestelmästä

Varastonhallintamme ja tilausprosessimme ovat aiemmin perustuneet voimakkaasti yrittäjän muistiin ja olemmekin tehneet monta päätöstä ”mututuntumalla” pohjautuen omiin muistikuviimme.

Muistikuvat eivät kuitenkaan koskaan voita oikeaa tietoa ja ymmärrystä tilanteesta ja tämän vuoksi *Resource planning* -työkalumme on auttanut meitä useammalla osa-alueella kehittämään toimintaamme entistä dataohjautuvampaan malliin. Ymmärryksemme sekä toimintamahdollisuutemme ovat kasvaneet merkittävästi seuraavalla neljällä osa-alueella.

Varastonhallinta

Ymmärryksemme varastosta ja erityisesti sen arvon kehittymisestä on kasvanut merkittävästi uuden työkalun myötä.

Aiemmin varastonkehitys on nojannut kuukausittaiseen kirjanpitoraporttiin, jossa eritellään ainoastaan kuukausitasolla varaston muutokset sekä ainoastaan varaston kokonaisarvo -tasolla. Nyt pystymme tutkimaan varastonkehitystä helposti eri aikajänteillä, jopa päivätasolla. Visualisoitu näkymä auttaa myös helposti hahmottamaan dataa hyvän numeraalisen raportoinnin lisäksi.

Nopea kokonaistilanteen seuranta

Näemme *Resource planningin* etusivulta helposti esimerkiksi kuluvan kuukauden myynnin, kateprosentin sekä palautusten osuuden kuukauden myynneistä. Tämä tieto auttaa meitä nopeasti tekemään päätöksiä tulevien olevien toimenpiteiden suhteen.

Esimerkkinä mainittakoon, että voimme nopeasti tutkia vaikkapa alennuskampanjan vaikutusta kokonaiskatteeseemme. Raportti eroaa myös sopivasti kirjanpitoraportin tuomasta tiedosta, sillä siinä missä kirjanpitoraportti mittaa tapahtumia tilitiedoista, mittaa *Resource planning* tapahtumia siinä hetkessä, kun ne tapahtuvat kaupassa. Ero näiden raporttien välillä voi olla merkittävä, sillä osa maksutapoja toimittavista yhteistyökump-

paneistamme tilittää saatavamme kolme viikkoa itse tapahtumasta. Kirjanpidossa tällainen tapahtuma kirjautuu myyntinä väärälle kuulle, kun taas *Resource planning* -raportista näemme oikean tapahtumakuukauden ja pystymme paremmin liittämään sen toimenpiteisiimme kuten vaikka voimassa olleeseen alennus- tai mainoskampanjaan.

Tuotekategorioiden seuranta

Tuotekategoriamme jakautuvat pääosin edustamiemme merkkien perusteella. Merkkien välillä on suuria hintaeroja keskimääräisessä jälleenmyyntihinnassa. Aiemmin arviomme tuotteiden merkittävyyttä liikevaihtoomme sekä katteeseen paljolti muistimme mukaan ”näitä on lähetetty viimeaikoina tosi paljon”. Valitettavasti tällä tavalla arvioidessa edulliset tuotteet korostuvat liiaksi ja arviomme on erittäin todennäköisesti väärä.

Tuotekategorioiden seuranta onkin osoittautunut oivaksi keinoksi luoda näkymä tuotemerkkien suoriutumiseen toisiaan vastaan. Vaikka edullisempi 85,00 euron hintainen tuote myy kappalemäärissä enemmän, tarvitsee niitä myydä 3,11 kappaletta, mikäli halutaan päästä samaan liikevaihtoon kuin 265,00 euron hintaisella tuotteella.

Tuotekategorioiden seuranta antaa myös erinomaisen näkymän kunkin merkin tuomaan katteeseen ja se auttaa näkemään, mitkä kategoriat todella tuovat voitot sisään. Visuaalinen toteutus auttaa helposti hahmottamaan trendejä esimerkiksi kuukausitasolla, jossa saattaa olla eri tuotekategorioiden välillä merkittäviä eroja myynnissä.

Yksittäisten tuotteiden seuranta

Koska yksittäisten tuotteiden seuranta on aiemmin ollut muistinvaraista, on ongelmaksi huomattu huonojen tuotteiden varastoon jääminen. Uudelle työkalulla pystytään tunnistamaan huonosti kiertävät tuotteet ja toimimaan saadun tiedon pohjalta.

Aiemmin yksittäinen tuote on saattanut hautautua varastoon ja jäädä sinne ilman, että kukaan on huomannut mitään. Nyt voimme aktivoida tällaiselle tuotteelle esimerkiksi alennuksen ja koittaa saada sen kiertämään kassaan, jotta voimme taas hankkia paremmin suoriutuvia tuotteita.

Tuotteiden ABC–analyysi on erinomainen tapa verrata yksittäisten tuotteiden suorituskykyä erilaisilla aikajaksoilla ja tunnistaa tärkeimmät liikevaihtoa sekä katetta tuovat tuotteet. Toisaalta voidaan tunnistaa tuotteista häviäjät, jotka pitää saada nopeasti liikkeelle pääomia sitomasta.